2012.10

PROGRAMMER 程序员



创业·求生

20 拥抱技术变革
2012 中国软件开发者大会报道

60 精益创业和敏捷

63 易用性将成为未来的主要角逐点
专访又拍云存储 CTO 沈志华

73 手机游戏付费模式分析

77 揭开ipa的面纱
简述App逆向分析工具与过程

90 云数据库服务的架构探索
MySQL

94 开源IaaS云平台分析与比较

104 腾讯分析系统架构解析

112 Hash函数与带有防欺骗的承诺

116 Node.js 的核心与红利

120 前端模块加载策略

130 小鳄鱼之父
Bart Decrem

ISSN 1672-3252
9 771672 325128 10>
邮发代号：2-665　定价：15元
www.programmer.com.cn　www.csdn.net

Contents



P20

资讯

7 外刊速递

10 网文精选

12 新闻

14 新产品新工具

16 程序天下事

20 拥抱技术变革
——2012中国软件开发者大会报道

22 改变世界的敏捷
——Agile 2012美国达拉斯大会回顾



P28

封面报道：移动互联网 创业·求生

移动互联网蕴含着巨大财富，但与此同时，创业之路却举步维艰：产品同质化竞争加剧，推广成本不断攀升，盈利模式迟迟未能清晰，投资者忽冷忽热……究竟要跨越多少道坎，我们才能生存下去？

本期封面报道，就移动互联网创业路上的关键“痛”点，我们邀请了前沿观察者、投资者、渠道商、数据平台以及亲身实践者们，分享自己的思考与心得，帮助更多创业者们在求生的十字路口做出更好的选择！

无论如何，“活下去，才有希望”。

29 2012，当我们谈论移动互联网创业时，我们在谈论些什么？

32 移动互联网创业路向何方？

36 移动互联网创业回归内容本身

38 2012移动开发者经济学

40 颠覆Craigslist的机会与陷阱

42 游戏创业团队的技术选型

48 靠谱的渠道在哪儿？

52 如何走向海外？

56 走出迷局，让用户关系沉淀
——陌陌科技运营总监王力专访

58 移动创业，有梦想就有希望
——UC产品总裁何小鹏专访

Contents



管理

60 精益创业和敏捷

本文结合热销图书《精益创业》中的核心观点，清楚地阐释了精益创业和敏捷之间的内在联系，并在文章结尾得出结论：精益创业和敏捷开发双剑合璧，才是创新型企业成功进阶的利器。

63 易用性：未来的主要角逐点

——又拍云存储CTO沈志华专访

66 产品营销

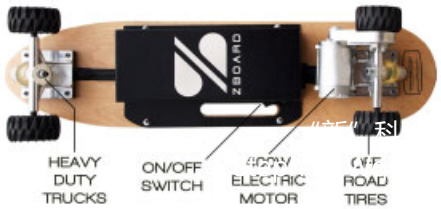
69 如何有效地控制项目进度

能否有效地控制项目进度是决定项目成败的关键，也是项目经理工作能力的重要体现。本期三位嘉宾将结合自己多年的实践经验，与大家共同探讨有效控制项目进度的技巧。

移动







72 简单的设计

73 手机游戏付费模式分析

移动游戏无疑是现阶段最赚钱的应用类型，但其盈利模式可不止一种。本文对当前的收益模式进行划分，详细探讨了在不同的情形下应该选择怎样的付费模式。

77 揭开ipa的面纱（上）

——简述App逆向分析工具与过程

82 Android软件安全开发实践

本期从组件间通信、数据验证和保全保护等方面来实践Android软件安全开发之路。

86 Android敏捷开发指南

云计算

90 MySQL云数据库服务的架构探索

虽然近两年来NoSQL发展迅猛，但MySQL凭借其成熟的中间件和运维工具及良性的生态圈，在互联网企业中的应用仍然非常广泛，并占有主导地位。本文将深入探讨依托MySQL平台设计和实现的UMP系统的架构探索经验。

Contents

主管：中国社会科学院
主办：中国社会科学院文献信息中心
出版：《程序员》杂志社
网址：http://www.programmer.com.cn
国际刊号：ISSN 1672-3252
国内刊号：CN11-5038/G2
邮发代号：2-665
广告经营许可证号：京东工商广字0188号

总编：黄长著　Editor-in-chief：Huang Changzhu
社长/常务副总编：张悦校　President：Zhang Yuexiao
副社长：蒋 涛　Vice President：Jiang Tao
编委会：黄长著　张悦校　陈洋彬　蒋涛　曾登高　刘江
Editorial Member：Huang Changzhu　Zhang Yuexiao　Chen Yangbin
Jiang Tao　Zeng Denggao　Liu Jiang

执行总编：刘江　Executive Editor-in-chief：Liu Jiang
执行主编：孟迎霞　Managing Editor：Meng Yingxia
编辑部主任：董世晓　Director：Dong Shixiao
责任编辑：陈博　杨爽　卢鹄翔
Editors：Chen Bo　Yang Shuang　Lu Dongxiang
特邀编辑：蔡学镛　池建强　冯大辉　高博　肖新光　杨栋　余晟　周爱民
Contributing Editors：Cai Xueyong　Chi Jianqiang　Feng Dahui　Gao Bo
Xiao Xinguang　Yang Dong　Yu Sheng　Zhou Aimin
美术设计：纪明超　Art Designer：Ji Mingchao
美术编辑：朱凯　Art Editor：Zhu Kai
流程编辑：白羽中　Coordinator：Bai Yuzhong
Tel：010-64351458
E-mail：editor@csdn.net

广告总代理：北京创新乐知信息技术有限公司
Sole Advertising Agency：Bejing CSDN Co.,Ltd
Tel：010-64376055
E-mail：ad@csdn.net
Marketing Dept：010-51661202（ext 149）
E-mail：market@csdn.net

发行部
Distribution Dept.010-64351431
E-mail：reader@csdn.net

读者服务部
Readers service Dept.
网上订购：http://dingyue.programmer.com.cn/
读者信箱：reader@csdn.net
地址：北京市朝阳区广顺北大街33号院1号楼福码大厦B座12层
Address: B-12th Floor Fairmont Tower NO.33 Guangshun North street,Chaoyang District,Beijing
邮政编码：100102
电话：010-64351436
传真：010-64348545

法律顾问：北京中润律师事务所 王杰
Law Consultant：Beijng Zhongrun Lawyer Firm
印刷：北京盛通印刷股份有限公司
Print: Beijing Shengtong Printing Co., Ltd.
出版日期：每月1日
Publication Date：the first day per month
零售价：RMB 15.00元　新台币 390元　HK $ 35.00（港、澳）
US $ 9.00（海外）
Retail Price：RMB 15, NT$390, HK $ 35.00,US $ 9.00

本刊文章版权所有 未经许可不得转载
发现装订错误或缺页，请将杂志寄回本刊读者服务部调换

94 开源IaaS云平台的分析与比较

99 Trident Storm与流计算经验

104 腾讯分析系统架构解析

TA（Tencent Analytics，腾讯分析）是一款面向第三方站长的免费网站分析系统，在数据稳定性、及时性方面广受站长好评，其秒级的实时数据更新频率也获得业界的认可。本文将从实时数据处理、数据存储等多个方面带你深入探寻TA的系统架构及实现原理。

108 HipHop算法
——利用微博互动关系挖掘社交圈

技术

112 Hash函数与带有防欺骗的承诺

互联网给我们的日常生活带来便捷的同时，也带来了信息安全隐患。很多安全问题可以简单地利用物理手段解决，但在纯信息交流层面，这可能会困难许多。在上期杂志中，我们看到了“秘密共享的门限方案”及其应用。本期让我们来研究一下带有防欺骗的承诺。

116 Node.js的核心与红利

120 前端模块加载策略

随着前端开发技术的蓬勃发展，需求大量移往前端处理、甚至将网页做成类似Gmail、Google Calendar等交互型应用程序。这个趋势使得CSS与JavaScript的代码量大幅增加，也使得加载许多现成的“库”来满足复杂的需求。本文将结合工具及范例，介绍加载JavaScript和CSS的数种方法与模式，进而引出载入前端模块的理想方式。

125 SMSS离奇被杀之谜（上）

百味

130 小鳄鱼之父Bart Decrem

132 图书

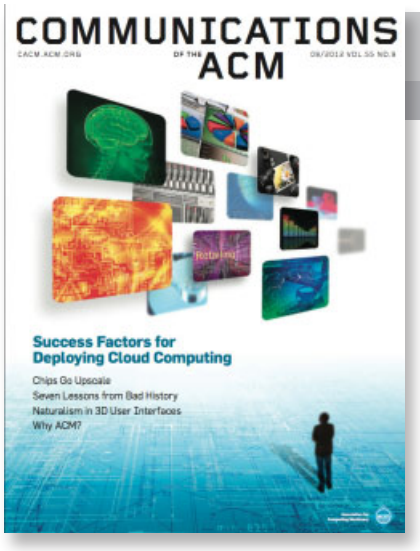
134 GEEK

136 幽默

企业专栏

24 跨平台数据库调优利器 DB Optimizer

26 “随手记”：借力百度云开发者平台获取资源和信心



Communications of the ACM

2012.9

三维用户界面之问

几十年来，以按钮、键盘、鼠标和操作杆为基础的用户界面（UI）不仅主宰着常规计算机领域，而且占据着更为专业的视频游戏世界。不过，近年来随着设计者迎来用户界面的空间输入，变革渐现。这些界面名目众多，包括姿势型界面、动作受控型界面、直接型界面、控制者隐退型界面、自然型界面。不管名目如何，总的特点是：在物理意义的三维环境下采用空间输入，我们将这种三维环境称为三维用户界面（3D UI）。主机游戏系统三维输入是最广为人知的三维用户界面实例，手机、平板电脑等移动设备也加入了这种趋势的行列。

研究发现，这些高配置三维用户界面几乎无一例外地尽量通过三维互动让界面显得更为自然。我们将用户界面的自然主义（或者叫做互动逼真性）定义为在用户界面中完成某个任务的动作与在现实生活中完成该任务的动作相契合的客观程度。摆动手臂打高尔夫球显然比按动一连串按键完成同样的任务更为自然，因为前者通过头部和身体的动作观察环境的特定部分，就像在现实中的做法一样。当然，自然主义水平同时取决于互动技术和任务环境两者——如果要完成的任务是驾驶虚拟汽车，那么方向盘隐喻是自然的，可如果要完成的任务是朝虚拟篮筐投篮，则方向盘隐喻并无自然可言。

这类事例引发了关于自然主义与三维用户界面的一些有趣的问题：三维用户界面天生就比传统用户界面自然吗？设计者应以追逐三维用户界面的高水平自然主义为首要目标呢，还是另有其他更为重要的互动设计标准？更自然的界面会带来更好的性能，还是更好的用户参与性，抑或更容易学习？如果无法使用最自然的技术，那么是用中庸的自然技术，还是用传统的技术更为合适？

维吉尼亚理工大学的一支研究团队通过一系列实验的大量任务，针对互动逼真性水平给用户表现、存在感、职业及喜好所带来的效果进行了研究。部分研究会调整显示逼真性水平，因为研究者假设所使用的显示类型会对互动逼真性有影响。这些研究结果为前面提出的问题带来启示，表1总结了针对各种用户任务的研究结果。

三维用户界面自然互动的影响力还有待发现，为了确定高度互动逼真性的优点，还需要探讨更多任务和应用语境。有一个重要问题是，互动逼真性不是二元的。简单地说，一个用户界面自然而另一个用户界面不自然是错误的。互动逼真性各种组件的影响还需要更多实验进行探索。

芯片再上层楼

50多年来，半导体芯片基本上都是扁平的，布置在其中的晶体管压得越来越严实。对摩尔定律的追逐使得晶体管越做越小，在平面上的排布越来越紧密。今天，耗电管理、散热及制造成本的上涨放慢了这种小型化的脚步。

芯片的制造过程大致是这样的：将电路设计投射到一块扁平硅表面上，通过光电化学过程将电路蚀刻在硅片上。可是，随着电路框架越来越小，这些电路

表1 特定用户任务下自然三维互动的优点和局限性

任务	自然主义的优点	自然主义的局限性
观察点旋转	用户更喜欢身体旋转	相对于身体和虚拟旋转同时进行，用户更喜欢虚拟旋转
	进行视觉搜索时，自然旋转技术相比虚拟旋转表现更好	
观察点转换/运动	头部追踪可以改善空间理解和详细的空间判断	头部追踪的优点可能取决于其他因素，如立体显示
		中庸的自然技术可能会比传统技术性能差
操纵	自然技术改进复杂操纵技术的性能	高度自然的技术具有有限范围
	高度自然的技术能提高用户能力	高度自然的技术往往降低精确性
	有可能设计出高度自然的技术，带来自然感以及拥有高度的精确性	中庸的自然技术可能会比传统技术性能差
车辆方向控制	提高互动逼真性可能令客户感到更有趣	中庸的自然技术可能会比传统技术性能差
瞄准	高度自然的瞄准技术可能比基于鼠标的技术表现更好	
多任务	高水平的互动逼真性如果与高度的显示逼真性相结合，会产生非常好的性能	如果整体界面不熟悉，高度自然主义可能不会有好处
	用户感觉高度自然的技术更令人投入，能带来更高程度的存在感	

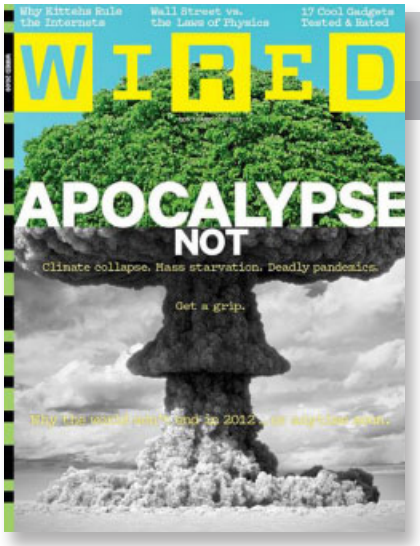
开始“漏”电流，因此既耗电，又发热。尽管无人知晓大限所在，不过传统硅晶体管的瘦身过程似乎即将到头。

芯片行业为耗电量和成本问题找到了答案——3D，让电子设备纵向发展，而不是横向发展。虽然原理简单，甚至可谓显而易见，但要做到经济可靠的生产实属不易。3D概念变型丰富，基本方案却只有三种，通常分别称为：3D封装、3D集成，以及单片集成。斯坦福大学集成电路专家Simon Wong教授认为，尽管目前3D芯片产量有限，但十年以内将在市场上得到普及。

“我相信摩尔定律还能应用好几代，因为还会涌现各种针对减小中间功率损耗的创意”，Wong教授说：“不断上涨的制造成本是一个问题。利用3D，可以持续提高性能，减小每一个技术节点的功率损耗，将每一个节点的寿命延长到3年以上，制造商因此有更多时间收回高额投资。”

与其继续给晶体管瘦身，不如设计一个三位系统，使其在相同程度上改进性能，Wong教授认为3D芯片用途广泛，尤其对移动设备特别有利，因为移动设备空间有限。

最古老、最简单的3D封装方式是将两片芯片面对面布置（称为“倒装芯片”），通过焊料“碰撞”进行连接。更先进的多芯片焊接理念则将传统2D芯片进行多层堆叠，然后在边缘进行电焊连接。这种方法相对较为便宜，采用的是传统半导体方法，但最终产品却十分粗笨，因此不适合小巧的移动设备，且速度相对较慢，因为信号必须通过多层芯片。目前消费者常见的闪存已开始采用这种堆叠方式进行生产。



Wired

2012.9

帝国反击战：英特尔进军手机市场

在计算行业遭遇iPhone、iPad、Android手机和平板电脑劫持之前，英特尔公司彻底控制着芯片帝国，以至于美国联邦贸易委员会、AMD公司、纽约州、欧盟及韩国都不约而同地针对它提起反垄断诉讼。但突然之间，消费者改了主意，他们不再需要能带来最前卫三维游戏体验的大功率机器，转而开始追逐运行小型应用程序、无电池耗尽之忧的便携式节能设备。

遗憾的是，英特尔的竞争对手知道如何制造适合这些便携式设备的芯片。二十世纪八十年代中期，英国Acorn计算机公司创造了后来名闻遐迩的ARM架构；1990年，苹果公司与Acorn公司合作，创造出用于苹果公司首款PDA——Newton的芯片。两家公司将ARM剥离出来，成立了一家独立的公司（苹果公司在九十年代售出了该公司的股份）。ARM一直支撑到现在，迎来了智能手机。如今，市场上95%的手机（包括iPhone和几乎所有Android手机）都用上了ARM芯片。

英特尔就此身陷危机。计算技术的未来在于体积小巧、电池供电。智能手机和平板电脑横扫传统PC市场，计算机芯片制造商要么分羹其中，要么静待淘汰。与此同时，开发商编写的成百上千个应用程序只能用于ARM指令集的iPhone、iPad和Android设备，英特尔处理器对这些程序束手无策，所引起的网络效应眼见要将英特尔挤出手机市场。

2010年，Paul Otellini聘用Mike Bell，他受命创造可与ARM一较高下的低功率芯片。Bell接过重任后不仅要创造低耗电芯片，还要创造能容纳这种芯片的手机，这是个庞大的工程——软硬件集成、测试、认证，对于少有手机开发经验的芯片公司来说尤其艰难。Bell从老东家苹果公司挖人，从Palm公司挖人，从竞争对手Qualcomm公司挖人，他聘请大批Android软件开发人员调试Google开源手机操作系统，令其能在英特尔手机上运行。Bell入职一个月后，英特尔公司斥资14亿美元收购德国芯片制造商Infineon Technologies无线部门，该部门的通信芯片采用ARM架构。随后，Bell让手下迁入距离英特尔公司总部几英里远的一幢没有标志的大楼，备好食物、饮料、床位，又迅速挂上了海盗旗。

一年后，他们造好了第一只工程样机。2012年1月，英特尔在CES国际消费电子展上展出了这款手机。英特尔手机还不是iPhone的对手，毕竟这才是英特尔手机第一轮面世，不过，它已独具价值，Orange的San Diego手机在英国售价为300美元，而三星或HTC的同类产品售价则为500~600美元。



Technology Review

2012.9

35岁以下的互联网创新者

MIT《Technology Review》杂志开列了一份“35岁以下创新者名单”，这些年轻的创新者来自世界各地，有的为消费者开发出各种网站服务，有的做出了有待商业化的更根本突破，不少人在前无古人的领域试验开拓…… 他们有一个共同的重要特点：他们的事业有可能在很长时间内独领风骚。本文节选了其中互联网领域的创新者，希望他们的故事能够给读者带来收获和鼓励。

Daniel Ek，29岁

革新：发展付费在线音乐业务，却并不强迫人们分别为每一首曲子掏钱。

Ek的解决方案名为Spotify，这是一款云唱机，用户可通过它按自己的需求合法访问数百万首歌曲。在订户及仅向非订户播放的广告支持下，这项服务在欧洲运营了三年，并于2011年进入美国。Spotify如今拥有超过1500万用户，其中400万为付费用户。Spotify估价40亿美元，是全球最红的网络公司之一。

Ek的感悟：“不断向自己提出傻问题。”

Ben Silbermann，30岁

革新：设计巧妙的社交网络，分享图片和兴趣。

Ben Silbermann发现了其他技术公司疏漏的东西：现有社交网络允许用户分享信息，形式广泛，但未做到从情感上、视觉上呵护用户的激情。怀着这样的想法，再加上在搜集方面兴趣浓厚，Silbermann和他手下的工程师创建了Pinterest网站。

Pinterest网站如今有60多名员工，在旧金山拥有宽敞的办公场所，所得到的风险投资总计达1亿3千8百万美元。到最后一次注资时，公司价值15亿美元。Silbermann说他对改进产品的关注超过对赚钱的关注。不过，零售商却发现，他们可以通过鼓励人们分享产品图片在Pinterest上推销产品。看来Pinterest公司可以考虑向零售品牌收费。

Silbermann说过：“帮助人们发现他们从前不知道自己想得到的东西”，价值就在其中。

Shishir Mehrotra，33岁

革新：创建可选广告，将在线视频转变为赢利业务。

2008年，Shishir Mehrotra入职YouTube，掌管广告业务，当时如日中天的视频分享服务每日可迎来数百万浏览者。广告门外汉Shishir Mehrotra想出了这样一个方案：可忽略式广告，即广告投放者仅需为观众看到过的广告付费。Shishir Mehrotra赌对了，分析师估计YouTube 2012年的毛利将达36亿美元，与视频内容合作伙伴分享广告收入后，还能留下24亿美元，几乎是视频流业务网站Hulu 2011年广告和订阅揽收的六倍。这意味着Mehrotra帮助YouTube的东家Google解决了一个让许多快速成长的网络公司头疼的问题：如何从免费使用其服务的广大受众赚钱。

Christopher Soghoian，30岁

革新：出于对在线不良隐私政策的反对，他敦促相关公司改变做法，并在这些公司拒绝改变时发出警告。

Soghoian具有举足轻重的计算机科学家身份，Do Not Track（不得追踪）机制正是在他的帮助下获得开发。通过这个机制，人们可以防止网站追踪他们的在线行为。不过，作为黑客，他的大部分工作更多依赖的却是记者们的技术：整理《信息自由法案》问卷，或者以啤酒为诱饵联系各公司律师及华盛顿的国会助手。当他嗅到网上的安全漏洞和隐私缺陷，就会敦促相关公司——Google、AT&T以及Dropbox等停止使用那些置消费者于险境的隐私政策。如果这些公司拒绝改正，他将公开这些问题，并力求立执法者予以制裁。

（感谢译者李芳支持）

最中肯的建议

文 / Rob Pike　链接：http://bit.ly/U8I5UJ

Pike是《The Unix Programming Environment》及《The Practice of Programming》的作者之一，也是UTF-8的合作开发者。这里要讲的是他从Ken Thompson那里得来的，自己认为最为中肯的一条编程建议。

在加入贝尔实验室大约一两年后，我曾与Ken Thompson结对编程，为一款小型的交互式图形编程语言开发一个即时编译器。我打字快，于是操纵着键盘，编码时Ken就站在我身后。工作进展得很快，问题也源源不断。

每次出现问题，我总是本能地钻进去，开始检查栈回溯，插入打印语句，启动调试器……而Ken只是站在那里思考，完全无视我，无视我们刚刚写下的代码。

一段时间后我觉察到一条规律：Ken常常能先于我找到问题的原因。他经常突然宣布："我知道哪里错了。"多数情况下，问题还真就是他说的那样。

我明白Ken是在大脑中建立代码的模型，如果哪里不对，那就应该是模型中的错误。通过思考问题如何发生，他能直觉地感知模型中出现问题的地方，或者代码与模型不一致的地方。

Ken让我明白了调试前先思考的重要性。一头扎进Bug中，只能修正代码的局部问题，而首先思考一下Bug出自哪里，却能发现并纠正代码中层次更高的问题，从而改善设计，防止更多Bug。

我承认这在很大程度上是个风格问题。有的人坚持对一切问题做工具驱动的逐行调试。但我现在相信，不看代码的思考才是最佳的调试工具，因为它能带来更好的软件。

软件开发团队的扩展

文 / Adam Wiggins　链接：http://bit.ly/O6pObc

业务壮大时，我们对于如何扩展Web服务器、数据库及其他软件系统相对熟悉。然而，如何扩展开发团队是个同样重要的挑战。且听本文作者对Heroku团队不同阶段面临问题及其解决方法的分析。

第1阶段：家庭作坊

团队成长之初，2~4名开发者在一处非商用的场所工作。沟通和协调非常简单，几乎不需要管理。每个人都是全能的通才。每个人的脑子里也都装得下整个公司和产品的全部状态信息。

这一阶段，你是在创建并摸索一款具有最低可靠程度的产品，或者说摸索自己到底要做什么。这时任何组织结构或过程都会带来害处。

第2阶段：增募兵马

你得到了一笔资金，雇来更多开发者，规模达到5~9人，这时你会发现随意式的协调机制慢慢不灵了。沟通可能太多（跟踪6个人的工作真不容易），也可能太少（不同的人在修同一个Bug而彼此不知）。

这时你就需要增加组织结构了，但只需一点点。例如周一做做迭代计划，每天站立会议一下，用白板或小型工具跟踪一下大的任务项和Bug。但不要急于引入过多的组织结构和过程。随意一点，团队仍在正常运转，团队中会产生自发的领导，他们能就许多工作提供指导，同时自己也在写代码。

你也需要继续摸清公司的真正方向。此时引入官僚机制几乎必然会妨碍你真正想做的事情，须知在你探寻可扩展业务模型的过程中，那才是核心的东西。

这一阶段的关键是专注。每个人仍是通才，但整个开发团队在每一个小阶段上应该有统一的目标。许多公司都不是因为机会太少饿死的，反而更容易因为机会太多消化不良而撑死。

第3阶段：拆分团队

开发团队达到10~15人的规模，便到了组织结构大变化的边缘了。这么多开发者，什么事情都挤在一起只会失去目的性和方向感。写程序时，当一个类或源文件越来越大，解决的方法是拆分它们。扩展一个开发团队也是一样的。

合理的团队划分，关键在于找到定义明确的职责范围，以及团队间清晰的接口。对于它所负责的那部分产品，团队应该有自己的愿景和方向，有最大限度的自治，而无须从其他团队获得行事的许可及所需的信息。

此时，你应该已找到了产品和市场的结合点。

John Carmack谈软件工程的艺术与科学

文 / Andrew J. Ko 链接：http://bit.ly/SPTnPU

本文是作者根据John Carmack一次访谈的内容整理而成，在访谈中，John提出软件工程实际是一门社会科学。

在计算机学科中，真正谈得上科学的只有算法，“优化”也只能算工程。然而，真正花在算法及其优化上的时间在编程中所占比例很小。90%程序员所做的只是将功能需求以顺序、分支这样的程序形式实现出来。在这方面，我越来越不觉得有什么客观的方法能带来更好的软件。

自然科学的根本在于测量和重现，在于估计和验证。软件工程中除算法和优化外，其他与此毫无关系。相反，它们只跟程序员之间，甚至同一程序员在不同时间，自己跟自己之间的社会化交互有关。

程序员总在不断犯错，这已是老生常谈的事实。问题就在于，如果一种错误的做法能以正确的语法形式输入到程序中，那它就肯定会进入我们的程序。因为这个，我本人对静态代码分析极度热衷。因为我们总在不断犯错，我宁愿使用编程语言的一个更为严格的子集，并对程序员写的代码施以更加严格的限制。

最近一段时间，我开始坚持每天做一点代码复查。每天看看提交的代码，找点有启发性的问题给团队讲一讲。有些问题，例如最大限度地保持参数的常量性，相对客观，争议较少。但也有很多东西更像风格问题，虽然多年来我发现它们不断地带来麻烦，但仍有很多人说：“我没见过这样的问题，对我来说这不是问题，我也不会犯这种错误。”因此，能够找到一个实际例子来指出“这就是由它引发的问题”，真是件好事。

意识到软件工程并非科学之后，我希望找到更好的方法来对付人性的弱点。然而，考虑到铁打的团队流水的开发者，想用一致的方法培训团队谈何容易。应对程序员犯错的方法肯定有好坏之分，但很难量化。而且代码会存在很长时间，将有成百上千的程序员看到一段代码，使用它并以某种方式与之交互。

除了通过静态代码分析实施严格限制外，在软件API设计的层面上还有规模更大的问题，艺术的、技术的都有，我希望能在这方面找到更多可量化的方法。

培育黑天鹅

文 / Paul Graham 链接：http://bit.ly/S0ur5N

文本作者系Y Combinator联合创始人，这篇新作讲解了创业投资中的反直觉特点和黑天鹅效应。

这些年来我做过各种不同类型的工作，可我真不知道还有什么工作能像创业投资这样反直觉。要理解创业投资，最重要的两点是：

1. 实际的回报来源于少数几个大赢家；

2. 最好的想法起初看上去都是不好的。

第一条，道理我很早就懂，但直到发生在身上我才真正理解它。我们投资的公司总价值在10亿美元上下，但其中有两家，Dropbox和Airbnb，加起来就占了其中的四分之三。

培养相信仪表而不靠感觉的意识，这是飞行员训练中的重要一环。想在一个反直觉的领域取得成就，你也要像飞行员一相关闭自己的直觉，去做那些理智上合理的事情，即使感觉上那是错的。

财政意义上的成功只有一种：你投资的创业公司成了大赢家。否则，就算它“成功”，带来的回报也寥寥无几。然而，创业公司成为大赢家并不是一个简单的常数概率。

难上加难

最好的创业想法起初看上去都是糟糕的，这进一步增加了创投的难度。如果一个好的想法显然是好的，那应该已被人实施过了。“看上去不好的想法”和“好的想法”，两者的交集就是创业公司的最佳位置。然而，“看不去不好”的想法中，绝大多数是真的不好。

岂料更难

除了上面的难题，更难的还在于找不到明确的迹象表明你的选择是对是错。即使选对了一个大赢家，也只有两年之后才能知道。一些可以度量的指标看似有用，却带有极强的误导性。例如募集资金的数量，这个跟能否成为大赢家毫无关系。我们要找的是那种离群的赢家，只要一个赢家带来万倍回报，我们便可以挑选千家公司，其他的一分不赚，我们也赚了10倍。

在创业股资中，我们永远谈不上基于回报去承担相应比例的风险。能够期望的不过是每当面对一帮看上去不错的创业者和一个疯狂的想法时，能够坚持说“管别人怎么想呢”。当初我们就是这么看Airbnb的，也只有让自己更擅于这种思考方式，我们才能发掘更多的Airbnb。

（感谢译者王江平支持）

桌面Linux路在何方

Linux在诞生之初便有机会在桌面领域占据一席之地，但至今仍无太多建树。GNOME创始人之一Miguel de Icaza日前分析了个中缘由。

Linux标准化的努力始终没有成功，几个发行版之间要么计划和步调不一致，要么重复实现相同的功能。每个人都想创新，想要创造Linux的下一个重大功能。假如你是领先的发行版，那么不兼容就是你的竞争力，是你获取市场份额的武器，你会自废武功吗？

Linux是软件包管理机制的先行者，但支持桌面Linux对很多独立开发者来说门槛非常高，他们开始叛逃至更优秀的平台，不仅是OS X，还有互联网。实用为先的最终用户也发现每六个月折腾一次声卡或显卡驱动不值得。很难想象App Store这样的生态系统有一天会出现在桌面Linux上。

Linus Torvalds不认同Icaza的观点，他认为Linux内核本身成功的原因之一恰恰在于他没有强烈的愿景，没强迫开发者遵循他的方式。至于桌面Linux不成功，那是GNOME项目自己的问题。

然而Torvalds的这条规则拿到桌面领域则很难灵验，想让消费者买账，一致的用户界面和强有力的推动者缺一不可。不相信？想想成功把UNIX交到用户手里的苹果，以及Google是怎样将Linux包装成Android并在移动互联网站稳脚跟的。



群英战地图

一名英国牧师曾写过本小书，名为《Flatland》，那里的居民都是线段，没有空间概念。想到某地只能一头撞下去，因为在同一平面，没有什么能提供指引。比妄想小说幸运的是，人们总有各种方式标识路径，移动时代手机上的地图更是极大地方便了出行。

Google Maps应该是最早引爆流行的地图应用，它的兴起不但带动了Ajax这样的技术流行，还带动了靠调用Google Maps API，以提供服务的众多应用，一时间，可谓风光无限。Google一出招，无疑挑动了众多对手，Yahoo!、微软等也推出了自家的地图服务。其时，另一位IT大鳄正在窥伺，只是时机还未成熟。

苹果2007年推出iPhone，内置了很多Google的应用。以苹果“卧榻之侧岂容他人鼾睡”的作风，这么做只因能力未及，不得不如此。2010年起，苹果开始一连串收购地图公司，继而在iOS 5中明言“地图数据”条款，终于在iOS 6中推出3D地图，在极短时间补齐与Google Maps的差距。同时，Foursquare、维基百科纷纷投诚开源地图，也让Google Maps遭遇了一番“众叛亲离”。

有人把Amazon比作屋里的大象，它无疑是互联网领域的重量级选手，却又常常被忽视。最近，它宣布与诺基亚合作地图服务，这使它摆脱了第三方的束缚，更得心应手地发展自己的软硬结合之道。

在《Flatland》里，居民们不得不摆动身体发出声音，以避免相互冲撞。而在互联网，战壕已经挖好，一场冲撞不可避免。

“在所有商家都接受智能手机付款之前，消费者无法舍弃钱包，否则就会出现部分商家不接受的窘境，若要达成这项目标，至少得花十年。”

——移动支付平台ZooZ CEO Oren Levy谈到移动支付产业的未来时认为，尽管智能手机能处理的事务越来越多，但想真正摆脱钱包和钥匙，还要耐心等待。

“许多全新PC都预装了木马，这就是为什么我每次购买新电脑后，都会自己重新安装系统。”

——微软研究人员发现，中国市场的许多全新PC在出厂前就已被安装了木马。被称为世界头号黑客的Kevin Mitnick在谈到这件事时做了以上表述。

“我一直希望拥有一台Cray，但如今GPU编程比在超级计算机上编程有趣多了。”

——John Carmack认为，为超级计算机编程的乐趣在于，处理并行矢量运算时带来的那种紧张感。在GPU上编程不但能获得同样的乐趣，还能结合纹理和视频计算。

“当从Loren Brichter手中买到Tweetie那天起，Twitter就已杀死了它，我最喜欢的客户端。”

——Instapaper作者Marco Arment认为，Twitter并不需要桌面版的Tweetie，甚至iPad版也不需要。这款红极一时的软件，距离它的上一次更新，已超过一年。

“Facebook的市值在90天内蒸发了500多亿美元。换个角度看，这比雷曼兄弟在申请破产前一整年中失去的市值还要多。”

——在谈到Facebook股价下跌的影响时，《纽约时报》认为这不只是投资者的问题，它也迅速在公司内部产生了有关其能不能留住和吸引技术人才的切实问题。人才是任何科技公司的命脉。

“我希望大家都能接受专利交流，让这些专利能被最大程度地应用到各项技术上。”

——在谈到苹果和三星的专利案时，Steve Wozniak认为专利之争会阻碍公司的创新基因，阻止技术的相互交流及进步。

“我们用CoffeeScript完全重写了Dropbox浏览器端的代码库，结果令人满意。”

——Dropbox工程团队在一周时间内重写了23000行代码，为了加快进度并避免出错，在这个过程中使用了Js2coffee、Jasmine、Selenium等工具。

CSDN十大资讯

2012年8月16日–9月16日

01 后乔布斯时代，苹果乏善可陈

作者 / 魏兵

苹果新品发布会如期召开，会上推出了新一代手机产品iPhone 5，对iOS 6的新情况进行了展示，并推出了新的iPod系列产品。但也许是因为之前被爆料出的信息过多，也许是因为这家失去领袖的公司确实减缓了创新的步伐，大多数人似乎并不如自己期待的那样满意。

02 直击Visual Studio 2012发布

作者 / 付江

iPhone 5发布的同一天，微软在西雅图召开了Visual Studio 2012发布会。每次Visual Studio的重大版本发布，都可以透露出微软的战略变化，云、移动、敏捷、项目管理、新的数据服务模型则是当前的重点。2012年底微软将发布Visual Studio 2012的第一个更新。

03 Google新增六度空间搜索

作者 / 张勇

六度空间理论指出，平均只需六个人就可以联系任何两个互不相识的人。这一理论之前已被用在LinkedIn等公司的产品中，Google推出的六度空间搜索，则为人与人之间建立关系提供了更便捷的工具。只要在搜索框中输入两个人名，便可计算出两人的关系链。

04 Google眼镜创始人：时尚女孩必备

作者 / 张宁

纽约时装周上，模特带着Google眼镜行走于T台之上。Google眼镜创始人Sebastian Thrun指出：谷歌眼镜特别适合那些整天泡在社交网络上的年轻女孩。此举意在打出时尚牌，进一步加强Google眼镜在大众消费阶层的印象，尤其是女性消费者。

05 《StarCraft》开发的荆棘之路

作者 / 王然

Patrick Wyatt曾是Blizzard副总裁，参与过《WarCraft》、《Diablo》、《StarCraft》等游戏的开发。本文从《StarCraft》开发的惨痛经历中吸取教训，反思了开发过程中的很多错误行为，实为开发者少走弯路必读之文。

06 解密移动游戏开发与Unity 4引擎特性

作者 / 杨依帆

王楠是一名身在德国的独立游戏开发者，他参与开发过多款热门游戏，《超级拍拍羊》曾获2011 年中国独立游戏节决赛提名。他以海外华人的视角对移动游戏的产品细节与开发技巧进行解密，同时分析了 Unity 4 在引擎升级后的种种新特性。



07 不想成为苹果，只想做Amazon.com

作者 / Romain Dillet

Amazon发布Kindle Fire HD之后，许多人将其与苹果对比。Amazon会成为谁、超过谁或者打败谁，一时间众说纷纭。TechCrunch记者撰文认为，Kindle的推出并不是向苹果发出竞争讯号。事实上，Amazon正在一条自己的路上全速前进。

08 程序员招聘新平台DeveloperAuction

作者 / 张祺

以往人才通过面试后，在最终谈待遇时才发现公司提出的条件远低于自己的预期，白白浪费了时间和精力。DeveloperAuction规定，程序员可以不接受Offer，但如果双方一拍即合，公司必须按照之前商定的最高薪酬履行协议，这种面试方式将大大提高就职成功率。

09 Windows 7市场占有率首超XP

作者 / 魏兵

2010年9月至今，Windows XP的市场占有率不断下滑，已从61.91%降至42.52%。与此同时，Windows 7的市场份额由17.64%上升到42.76%。另一边，Windows Vista的市场份额由13.75%下降至6.15%。这也使得苹果公司的OS X超过了Vista，占据7.13%的市场份额。

10 每个程序员都应该知道的8个Linux命令

作者 / Jay Fields

在职业生涯的某个时刻，你总会发现自己需要了解一些Linux方面的知识。这并不是说你该成为一位Linux专家，而是意味着，当需要使用Linux命令执行任务时，你应该能熟练完成。学会cat、sort、grep、cut、sed、uniq、find和less这8个命令就足以应对绝大多数任务。

最具争议的10个编程观点

你认为最具争议的编程观点是什么？C#专家Jon Skeet曾在StackOverflow上提出这个问题。这里列举最热门的前10条讨论。

01 业余时间不以编程为乐者，永远赶不上那些以此为乐的同行

即使最聪明、最有才华的人，若只是将编程作为工作，就永远成不了真正优秀的程序员。以编程为乐的人会在业余时搞些小项目，或者摆弄各种编程语言和编程思想。

02 唯一能放之四海而皆准的最佳实践，是“用脑子思考”

太多人喜欢追逐众多时髦技术，想方设法把各种方法、模式、框架用到不适合的地方。新技术和名人大牛的观点并不等于适用于实际情况。

03 依赖Google没什么错

这种言论肯定会让那些学富五车的饱学之士恼火。但重要的是能真正理解，并给出成功的编程解决方案，让客户和老板满意。

04 大多数注释是代码的糟糕重复

我们大部分时间是在维护其他人（或我们自己）所写的代码，而糟糕、错误、过时和误导性的注释也是代码中最令人纠结的因素之一。

05 XML被大大高估了

许多随波逐流的人没动过脑子就跳上了XML这条黑船。XML用于Web应用不错，因为它本来就是干这个的。此外的问题定义、设计思路应该尽量不用XML。

06 程序员生而不平等

经理往往认为“程序员A＝程序员B”，因为他们工作的年头差不多。但实际上，他们的效率可能相差十倍甚至百倍。

07 将Java作为最适合大学教学的第一门语言令人费解

第一门编程语言应该重在学习控制流和变量，而不是对象和语法。此外没有调试C/C++内存泄漏经验的人，根本无法完全理解Java的初衷。

08 若你只会一门语言，无论多么精通，仍然不够优秀

任何人如果只局限于一门语言，都无法充分发挥自己的潜力。而且缺乏求知欲和探索意愿，都不符合优秀程序员的特质。

09 性能真的很重要

尽管可以通过购买更强劲的CPU、硬盘和内存提升性能，但若这样想的人少一些，我们也许能在1GHz的CPU上运行更多程序。

10 Print语句是有效的调试方式

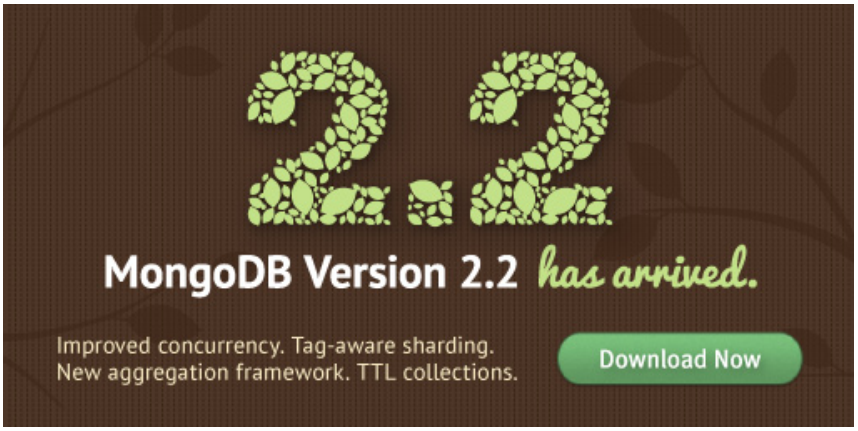
用System.out.println之类的输出语句调试代码的方式挺好。这经常比正式的调试要快，而且可以比较不同运行的输出结果。

MongoDB 2.2发布

MongoDB是高性能开源文档数据库，也是目前最受关注的NoSQL技术之一，以敏捷、可扩展和对企业应用友好而著称。有人甚至认为LAMP中的M应该用MongoDB取代MySQL，其火热程度可见一斑。使用MongoDB的公司包括Foursquare、Craigslist、迪士尼、SAP、Intuit、EA等。国内淘宝、大众点评、视觉中国等公司也有应用。

2.2版的改进与修补达1000多处，其中比较重要的包括：

- 增加聚集框架，现在不用MapReduce，也能轻松地操作和处理MongoDB中的文档了；
- 多数据中心支持，通过读优先级和带标签的分片，改善了MongoDB在分布在不同地理位置上的多个数据中心的支持；
- 改进了并发特性，主要包括：新增数据库级锁，新增一个新的子系统避免了在大部分页面错误情况下的锁定，并改进了Windows上的页面错误检测，同时还改善了并发报表。



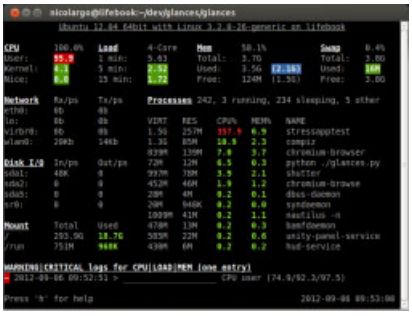
纯文本效率工具FoldingText

持久存储知识的最佳格式是纯文本，通过纯文本，我们给予了自己既能以手工方式，也能以程序方式操纵知识的能力。

FoldingText由TaskPaper的作者开发，是专为Geek设计的纯文本效率工具，它将流行的Markdown格式与大纲工具、任务列表完美结合了起来。在一个纯文本文件中，通过几个预定义的语法标记，文本将被自动格式为标题、列表、段落和待办事项。在保持简洁的同时，赋予了文档无限的扩展能力。95%的FoldingText代码使用了Web语言，为它在多平台上扩展打下了良好的基础。



命令行系统监视工具Glances



Glances是一款用于GNU/Linux、BSD和OS X下的开源命令行系统监视工具，它使用Python语言开发，通过PsUtil获得系统信息。Glances默认每秒进行一次刷新，但可以通过-t参数指定间隔时间。它能够监视CPU、负载、内存、网络流量、磁盘I/O、文件系统、进程和日志方面的信息。正常状态的数据以绿色显示，需要注意的信息显示为蓝色，品红和红色的数据则分别表示“警告”和“危险”。

在Glances运行时，按h键将显示帮助信息，按a键则依据当时的系统状况，自动将监控信息按照重要性排序。

Windows Server 2012发布

微软在中国市场正式发布了新一代云操作系统Windows Server 2012，以期提供跨越不同数据中心与云架构的现代化IT整合解决方案。Windows Server 2012首次采用面向海量计算和存储资源、虚拟机和应用程序的云设计理念，帮助用户构建跨越私有云、托管云和公有云的现代化IT架构。Windows Server 2012带来了超过200项创新功能，其中包括高级存储、网络、虚拟化与自动化功能等。

PostgreSQL 9.2发布

开源数据库PostgreSQL发布了9.2版，性能获得了大幅提升，并增加了对JSON的内建支持。

PostgreSQL 9.2支持水平扩展到64个核心，能够进行仅对索引的扫描，并降低对CPU电力的消耗，从而显著改善了大多数工作负载下的可伸缩性和开发者的灵活性。在垂直可伸缩性方面的改善让PostgreSQL可以更有效地应用大型服务器上的硬件资源。

其他方面的改进还包括：允许查询计划器为指定参数值生成自定义的计划，提升计划器通过内部索引扫描来使用嵌套循环，允许流复制从节点将数据转到其他从节点，允许Pg_basebackup对待机的服务器进行基准备份等。



Web包管理工具Bower

Web开发者往往需要手动管理各种库的依赖关系，Twitter工程团队希望能改变这种现状。

Bower是由他们开发的一款开源Web包管理工具。Bower在Git之上运行，它将包依赖模型以简单的可编程API形式暴露给外部的Sprockets、LoadBuilder等Build工具。开发者可以通过Bower快速加载图像、CSS、HTML和JavaScript资源，并自动处理依赖关系。

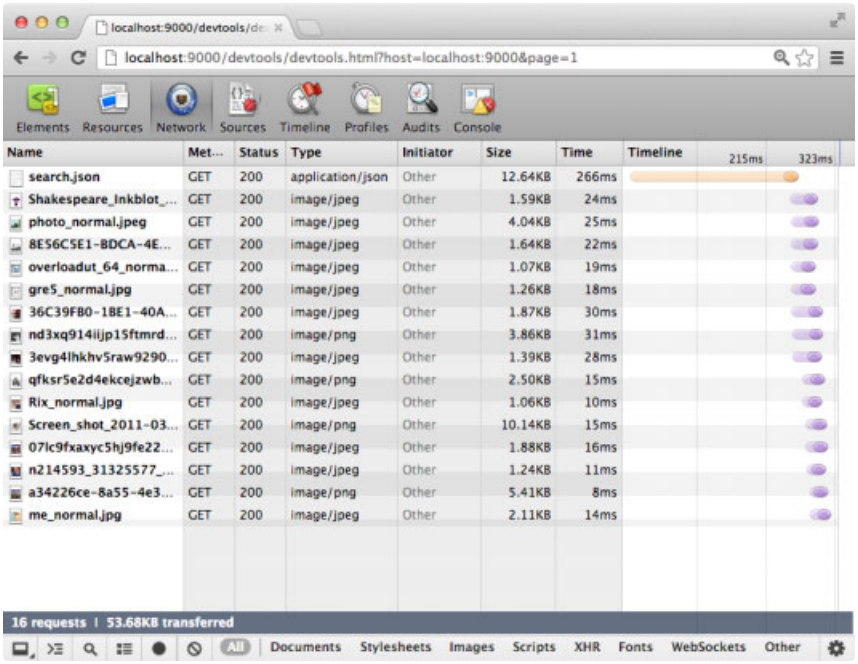
举例来说，运行bower install jquery命令，系统将下载jQuery并将其安装到./components/jquery目录。

iOS应用网络调试工具PonyDebugger

移动支付公司Square的iOS开发团队使用4个月的时间打造了一款用于iOS应用的网络调试工具——PonyDebugger。

PonyDebugger通过使用Chrome开发者工具调试iOS应用的网络流量和数据存储。同时提供一个iOS客户端用来调试通过NSURLConnection和其Core Data堆栈发送的数据。

其名称来源于Square内部的Objective-C网络库PonyExpress，开发这款工具最初的目的是为了能够对Square的网络流量进行远程调控。但随后Square决定将PonyDebugger扩展为适合所有iOS应用的调试工具，并将其源码公开。



具备GUI特性的Shell终端Xiki



Xiki除了可以做到其他Shell终端能够完成的任务，它还模糊了GUI与CLI的界线，在其中任意输入一个命令，例如git、bootstrap、mysql或mongo，都可以通过鼠标双击或者“Control+回车”执行。

Wiki中的任何内容都是可编辑文本，正是受此启发，Xiki中的所有文本均可编辑，也可以在任何地方输入命令或编辑输出。

目前Xiki可以以插件的形式在Emacs、Vim和CodeMirror中运行，其他编辑器的移植也正在进行中。

UDID罗生门

关于智能手机及其盈利模式对用户隐私侵害问题的讨论一直没有停止，手机制造商、OS厂商、应用开发商纷纷进入黑名单。

本月的热点事件是“Anonymous Leaks 1M Apple Device UDIDs”——著名黑客组织Anonymous宣布在AntiSec行动中入侵了FBI的电脑，获取并公布了一批苹果用户移动设备的UDID及其他信息。

Anonymous对事件的细节进行了如下说明：

在2012年3月第二周，Anonymous曾利用Java Atomic Reference Array漏洞入侵FBI区域网络行动小组和纽约FBI办公室的证据响应小组高级探员Christopher K. Stangl的电脑。在他参与会话过程中，一些文件从电脑桌面上一个名为“NCFTAiOSdevices_intel.csv”的文件被下载，这是一个包含12,367,232台Apple iOS设备的列表，列表内容包含：UDID（唯一设备识别码）、用户名、设备名、设备型号、苹果推送通知服务令牌、邮政编码、手机号码和地址等个人信息。

Anonymous公布了这100多万个UDID及对应的苹果推送通知服务令牌、设备名称和型号信息，以证明其确实获取了相关数据。并宣称其目的不是针对苹果用户展开攻击，而是为了让苹果用户或是更多人意识到FBI正利用这些信息追踪公民动态。

这个消息引发了轩然大波。

次日，FBI迅速对此事予以了坚决地否认，他们表示：“目前没有任何证据表明FBI有笔记本电脑被入侵，FBI从未收集过苹果用户的私人信息。”苹果也在次日声明从未向FBI提供过用户信息。

随后，佛罗里达州一家名为BlueToad的应用出版公司出面向NBC证实，被泄露的苹果设备UDID是黑客从他们的数据库中盗取的。该公司技术人员对比了泄露UDID与公司UDID列表，发现二者吻合度高达98%。

尽管此事看似可以结案，但我们还是能发现，相关的合理性推导还缺少必要的细节。除了这份已被公开的数据，目前各方均系自说自话。例如，有人质疑BlueToad是不是一个“顶包者”，因为BlueToad没有提供足够充分的细节。而即便BlueToad的数据比对结果是真的，也只是说明了数据初始来源；由于他们没有提供发生攻击和获取信息这一链条的电子证据和分析佐证，因此并不能说明Anonymous的数据就是从BlueToad直接获取的（即可能确实是BlueToad的数据，但Anonymous从其他地方攻击获取到了）。在这一点上，FBI也没有对Anonymous披露的细节进行回应，例如是否确实有这名探员、这个人自己如何做出解释等。

Anonymous的推导也并不严密：即使这些数据真的来自FBI的电脑，也并不能说明FBI就使用了这些信息；这份信息也可能是FBI调查某次网络攻击的物证，或者是FBI探员的个人行为等。因此，在没有更多细节和证据的情况下，此事只能是一场隐私罗生门。

最近一段时间，关于智能手机及其盈利模式对用户隐私侵害问题的讨论一直没有停止，手机制造商、OS厂商、应用开发商纷纷进入黑名单。自由软件运动领袖Richard Stallman则把矛头直指国家和政府，他认为手机可以让国家了解用户的位置，并随时进行跟踪。当他发现移动手机能跟踪携带者的位置后，宁愿牺牲方便也要保护个人隐私和安全。他更笃定地认为“现代移动电话的实现方式在道德伦理上是不可接受的，因为它们根本就是监视和跟踪设备”。



肖新光
网名江海客，安天实验室首席技术架构师，研究方向为反病毒和计算机犯罪取证等。微博：weibo.com/seak。

智能手机的半程冠军

当技术、创意的优势在前半程消耗殆尽时，移动设备的下半程比赛拼的就是实力和信念了。

马拉松比赛里有一个名词“半程冠军”，对于一项残酷的比赛来说，能够在半程保持领先，已非常不易，但多年之后，又有谁会记得半程冠军呢？2012年9月，智能手机马拉松赛的选手们到达了自己的折返点。跑在最前边的，自然是黑白两色运动服的iPhone；穿着五颜六色的Android兄弟们紧随其后，但又不敢离iPhone太近，因为他们要随时躲避iPhone扔过来的香蕉皮；一身闷骚蓝的Windows Phone远远地跑在第二集团的前头，但距离第一集团确实还有些距离。

当技术、创意的优势在前半程消耗殆尽时，下半程拼的就是实力和信念了。9月一系列的发布会，是智能手机厂商的战前动员，三星延续了以往的抢跑风格，率先发布了自己的Galaxy和Windows Phone 8新品牌Ativ。受到官司的影响，三星对于Windows Phone的重视程度明显加强了，但抢先发布的Ativ确实没有太多亮点，是台很朴实的机器。

接下来的Lumia 920，作为一台“照相机中的战斗机”，确实给我们带来了一些惊喜，无线充电、PureView拍照增强技术、超大的屏幕。虽然微软和诺基亚的同学们还是不会讲故事，但无法掩盖Lumia 920的王者之相。Windows Phone 7时代，一直缺乏一款旗舰手机，Lumia 900虽然不错，但缺少可以拿出来炫耀的功能。闷骚和显摆们，都能在Lumia 920上找到感情寄托。当然，旗舰手机只是为了拉高用户的期望值，重头戏其实在低配手机上。这次发布会饱受诟病的一点是，没有宣布价格和上市时间，这一点可能是源于信心不足吧，毕竟要看iPhone 5的情况。

iPhone 5的发布会，观众们看完后松了一口气——“肾保住了”。确实离开了乔布斯的Apple缺了点什么。当然，任何产品到了第五代，创新就变得越发困难了。但如果教主在，至少能让教众们相信iPhone 5依旧是最酷的，只留下一帮开发者为屏幕分辨率发愁。

接下来说说下半程会发生的情况吧，Windows Phone听到的都是好消息，无论是官司，还是iPhone发布会，其实1997年的微软不也是在刻意扶植Apple吗？只是风水轮流转，微软变成了困难户。当然，微软是最善于采取紧跟战术的公司了，领先者不犯错误，我就一直跟着跑，你一旦犯错误，我就反超。三个版本之内，微软能够从绝对劣势变成绝对优势。但微软现在最大的敌人就是他自己，这家官僚主义盛行的公司，总会在某些不经意的地方犯错误，从而招致巨大的损失。

Android很快就会变成专利屠宰场，公司不大没事，公司一旦大了，一定会招致专利的打击，但大公司拿走了智能手机领域的绝大部分利润，如果Android永远做不大，开发者获利的机会自然少之又少。加上目前已浮出水面的兼容性问题，Android将不再是个人开发者的乐土。

而iPhone呢，有一句话大概最合适：生于创新，也可能死于创新。



马宁

北京支点联游CTO，OpenXLive开发者。曾任微软亚洲工程院软件工程师，从事Windows Embedded Compact开发工作。加入微软之前，连续四年获得“微软最有价值专家”称号。

大数据产品的维保策略和服务定价该怎么做？

对大数据用户而言，如何引入大数据产品是关键一步；但站在企业的角度看，怎样能让大数据产品长期平稳运行才是关键。

我们参与技术会议时，经常能看到大型电子商务企业关于Apache Hadoop等大数据产品的介绍，似乎这些产品就像被驯服的烈马承载着巨量的访问一日千里，而且它们不会停摆。但事实上，这里很多案例是不可复制的，因为物力、人才（尤其是团队中的个别杰出成员）这些显而易见的因素很难同步，所以尽管大型电子商务企业可以做到，但这并不意味着当前及未来几年这些产品的潜在用户也能这样驾驭大数据产品。

目前，国内能够从事大数据产品维保和服务的第三方顾问公司寥寥，更多企业用户的大数据产品则是从原有数据库提供商（如IBM、Oracle和微软）及服务器和存储供应商（如DELL、EMC）处获得，由于在推广期，维保费用出现了两种极端：一种是几乎不需要额外费用，完全纳入原有产品的服务费用中；另一种是将大数据作为奇货可居的商品，将本来开源的产品结合特定需求打包后作为高价解决方案售出。两种迥异的定价不仅出现在竞争厂商之间，即便在同一个厂商也存在差异，这种现象具有其存在的合理性，原因如下。

- 前者往往是“锦上添花”的产品，即将大数据作为一个辅助功能添加到市场竞争较为激烈的既有产品中，但每个主要厂商都这么想的结果就是加剧了大数据产品的竞争，因此为了这个“锦上添花”加价就不够明智了。

- 后者往往是解决“大问题”的新特性，例如面向企业内部一直游离于内容管理之外的音频、视频、邮件等非结构化数据等的分析功能，通过在大数据基础上结合公开技术标准开发的各类新鲜解决方案，自然价格不菲。加之大数据周边开源社区一时还难以成气候，因此厂商自然在定价方面占有主动。

不仅在产品特点上存在差异，大数据产品的价值构成也可以细分为“服务器和存储设备”、“大数据管理软件”、“大数据分析软件”、“大数据应用系统”等几个领域。从产品的生命周期看，各领域的产品特点如下。

- 硬件部分虽然门槛相对较低且当前市场非常热络，但这部分市场已趋于成熟并将逐步走入红海竞争的阶段，因此为了维护长期稳定的收益，可以考虑做大规模、降低成本的方式，逐步将这部分工作标准化，最后形成售前、售后两头重的定价和维保策略。

- 随着技术的发展，大数据管理软件自身开源趋势比较明显，这个领域“前低后高”不失为一个明智之举，毕竟短期内普通用户、企业用户对于大数据产品的熟悉尚需时日，而这样的大数据产品一旦用上想撤下来很难，因此先将大数据引入客户的IT环境，然后在售后上适当提高定价也容易被用户接受。

- 目前已陆续有一批大数据分析软件厂商崭露头角，但除了那些隶属于传统数据管理软件企业的“正规军”，还有很大部分新厂商，对他们而言，凭借产品旺盛的市场需求，提高推广价并签署长期维保合同比较利于培养用户忠诚度、形成比较好的互动关系，也为后续自身技术发展不断注入阶段性投入。

- 大数据应用系统则是实际将业务处理与大数据产品结合的关键环节。目前看来，市场前景旺盛，短期内市场缺口较大，因此开发、维保费用即便偏高，用户也很难拒绝。此外，考虑到这个领域市场前景很大，因此是否具有谙熟客户行业知识并能够用大数据技术有针对性的解决实际业务的专家、顾问就成了构成竞争优势的主要因素，这部分定价要更多考虑成员的技术储备和实战经验。

对大数据用户而言，如何引入大数据产品是关键一步；但站在企业的角度看，怎样能让大数据产品长期平稳运行才是关键，维保和服务的定价需要依靠用户的业务要求进行裁剪。大数据技术是一系列边界模糊的概念和产品，但我们不能只看到它的昨天和今天，在试图引入之前，还要更多考虑它的明天。



王翔
软件架构师，主要研究方向为XML、.NET、领域设计和PKI应用。工作之余喜爱旅游、写作和烹饪。

开放平台开发之我见

我们还生活在悬崖边上，这就是基于开放平台开发者的现状。

新浪微博开放平台

目前基于新浪微博的开放平台应用多如牛毛，尤其Facebook的上市，更刺激了社会化营销、社交推广之类的名词和应用出现。就像2000年初，IT界出现EJB（Enterprise JavaBeans）、Web Service一样，是个公司都能融到钱，基于新浪微博开放平台的应用也着实火了两年。当然，2012年的形势急转直下，天上掉馅饼的日子不会再有了。

在技术架构和稳定性上，新浪微博开放平台没有问题。但在商业应用上，新浪的政策反反复复，相信很多第三方公司都中过招。

我们基于新浪开放平台推出了一款名为“淘推微博助手”的应用，专门给淘宝卖家做推广。其在新浪微博开放平台的审核，可以说是层峦叠嶂，一波未平，一波又起。刚开始时，审核两天就通过了。那时感觉天空一片明亮，我也通过自己的影响力，迅速通过新浪微博推广，几天内就有几百个卖家使用。但一周之后，我不小心修改了微博审核通过的名字，然后点击保存，结果又进入了新浪来源显示审核名单。自此我的噩梦开始了，两天后，新浪回复说审核失败，主要原因是不支持多账号群发，不能发布电子商务信息到微博，如果需要发送，则必须通过Email审核。我们不断发Email提起申请，却没有获得任何审批，应用也被冻结，初期的客户几乎全部跑光了。最后我将多账号群发的功能去除，再次排队，才终于通过审核。但通过标准API所做的应用，如何才能有竞争力呢？这个问题平台是不管的。虽然抱怨新浪，但它有一点做得比淘宝好很多，就是新浪允许在微博API开发的应用内，添加自己的广告，而淘宝是绝对不允许的。

开放平台开发个人总结

无论国内还是国外，开放平台对第三方开发者的开放程度将会越来越严格。当然，开放平台有可能在某一阶段，针对开发者推出优惠措施和非公开的API，有心的开发者就要抓住这个机会，说不定一下就发展起来了，这个要看自身的实力，外加一点运气。我对基于开放平台开发的总结如下。

- 无论Android、iOS还是国内的淘宝、新浪，基于开放平台的应用之间竞争异常激烈，已是红海中的红海。
- 基于开放平台的应用，只能当做生意做，不能当做事业做。
- 不要做（或只做）基础类应用，如微博定时发，因为平台迟早要抢这块生意。
- 应用要能自圆其说，平台数据只是数据来源之一。

无论是不是基于开放平台的开发，我们的应用，一定要构建自己的防火墙。我对防火墙的理解如下。

- 产品上有核心技术，形成技术壁垒。
- 资源上有优势，包括行业资源垄断壁垒和行业背景知识壁垒。

基于第三方开放平台进行开发，由于大家采用的都是标准API，因此在功能上，很容易造成雷同或相似，第三方开发者也很难脱颖而出。基于iOS和Android开放平台的应用也面临同样的问题。2011年，我基于淘宝开放平台，推出了一款叫做“易库进销存”的进销存软件，是针对淘宝卖家用于打单、发货、库存管理的一款卖家工具，它就面临同质化严重的问题——大家功能都差不多，而卖家觉得很难用。2012年，我就吸取这个教训，基于新浪开放平台推出的“淘推微博助手”增加了对用户很有帮助的“微博来源显示”功能。当然，这个功能需要新浪的审批，我们还是没有掌握自己的命运。

我们还生活在悬崖边上，这就是基于开放平台开发者的现状，先学会忍受吧，痛并快乐着，毕竟它能为自己带来大量客户。



邢波涛
北京新软孚信息技术有限公司技术负责人。关注SaaS管理软件和B2B、B2C电子商务的融合。

拥抱技术变革

2012中国软件开发者大会报道

记者 / 卢鹄翔



2012年9月8-9日，由CSDN主办的中国软件开发者大会（SDCC）在国家会议中心召开。“反映软件产业的技术与发展，推动深入交流”是SDCC的宗旨，这场一线开发者盛宴在准备阶段就从社区征求对议题设置的建议，希望能围绕软件开发的多个角度，尽可能全面地帮助开发者了解最新的技术进展，解答他们关心的实际问题。为了让听众能有机会与演讲人进行更直接的交流，多个论坛都安排了圆桌讨论环节，从Twitter和Netflix也都特意邀请了华人工程师，避免因语言而造成隔阂。

移动互联网、社交网络、海量数据都在深深地影响着我们的生活，软件正在改变世界，软件技术本身也正经历变革——云计算赋予了开发者巨大的能量，而开放平台则为用户与开发者建立起了生态系统。但在这些变革之中，也有不变的地方，大多数应用依然可以分为界面、逻辑和数据三层。

设计驱动开发，产品如何平稳进化

界面是用户与软件交互的窗口，成功的应用离不开优秀的设计。在“产品设计”论坛中，frog青蛙设计资深技术专员Markus Boelter谈到，工程师往往将设计师误解为艺术家，但事实并非如此，设计师真正的角色是“问题解决者”。他阐述了在需求不断变化的情况下，如何由设计驱动开发，让设计师、工程师和用户开始构思产品时就紧密配合。“工程师与设计师的工作其实密不可分”，Boelter表示，他本人曾是芯片设计工程师，因而深谙此道。

成功的应用同样离不开运营，尤其是在复杂的互联网环境中，须能应需而变。冯大辉以丁香园为案例，分享了垂直社区产品从“传统”的Web 1.0模式向“现代”一点的Web 2.0模式演进的过程，以及在这个过程中如何进行产品的改造与改进，阐释了如何解决产品老旧、稳定性差、架构混乱、访问缓慢等问题。他提到，改造垂直社区尤其需要留意的是，将会面对许多非深度互联网用户，需要考虑他们的习惯。

前端技术——应对JavaScript的新挑战

对于互联网应用来说，界面则交到了前端工程师手中。随着交互式应用的出现，以及越来越多功能需求移向前端，项目的代码量在增大，开发团队成员在增多。但由于JavaScript的设计过于仓促，导致语言本身缺少一些很重要的特性。其中一个缺失就是对模块化开发的支持，以往的代码组织方式已成为高效开发和维护的障碍。台湾miiiCasa前端团队主管蒋定宇和支付宝前端开发部高级技术专家王保平分别从各自的角度阐述了模块加载策略及共享方面的经验，给出了经实践验证可行的提高前端项目代码性能和可维护性的方案。

JavaScript应用越来越广，也带来了调试的挑战，Node.js可不可以调试？eval()出的代码可不可以调试？赵劼在演讲中给出了肯定答案，并现场演示了如何借助SourceMap调试上线时经过压缩混淆的JavaScript代码。

智能算法解惑

智能应用无时无刻不在影响着我们的工作和生活，这些应用的









逻辑层，同时也是构建智能应用的关键技术——智能算法却常常披着神秘的面纱。一方面，它们的潜在价值可以带来巨大的经济回报，以往商业公司不愿轻易示人；另一方面，几乎所有的相关技术都源自学术研究，倘若缺乏引领，则难以深入其中。智能算法论坛汇集了国内顶尖专家——腾讯首席科学家孙国政、华为诺亚方舟首席科学家李航、豆瓣网首席科学家王守崑等。他们不仅介绍了各自所在领域的最新进展，未来发展方向和挑战，还详细解答了听众关于推荐、广告、搜索领域，以及其中具体涉及的机器学习等具体技术疑问。

新编程语言

搭建任何应用都离不开编程语言，新语言的诞生往往是为了解决既有语言的缺陷，更好地适应新场景，或者与其他语言配合，提高开发或运行效率。

举例来说，PHP拥有丰富的扩展库和HTML混写能力，使其成为最流行的Web开发模板语言。但随着应用规模增长，以及业务逻辑愈发复杂，它的劣势也凸显出来。这时，这需要采用一些后端技术与之配合。39健康网的邢兴，讲解了如何将新兴的Go语言与PHP Web开发相结合。

面对前文提到JavaScript存在的种种不足，创新工场首席架构师蔡学镛介绍了Google新推出的Web开发语言Dart，包括它的目标、特色、用途，以及未来的发展；厘清了关于它的误区，并说明它与JavaScript存在的具体差异。在演讲结束时他总结道，尽管Dart已经显露出种种优势，但真正在生产中使用还须谨慎，建议开发者观察一段时间后再考虑是否采用。

Coding for Fun

SDCC上另一个激动人心的环节来自首次增设的编程马拉松——Coding for Fun。活动中，100名开发者相聚在一起，在32小时内自由组建团队，进行产品策划、开发，最后向公众展示作品。除了技术的比拼，Coding for Fun考验的还有选手们的创意、合作和毅力。大概只有亲身经历才能体会到这32个小时过程中的辛苦与喜悦。

在大会致辞时CSDN创始人蒋涛说道：“我们迎来了一个最好的变革时代，但这个时代的发展速度飞快，窗口期也许只有两三年。”这变化的速度也能从大会的议题中窥见一二，开发者们热烈讨论的话题已很少与上一届存在交集。这似乎告诉我们，只有不断学习，拥抱技术变革，才能乘着浪尖前进。



改变世界的敏捷
Agile 2012美国达拉斯大会回顾

文 / Shining Hsiong　图 / Tom Poppendieck

今年已是我第二次参加由Agile Alliance举办的Agile Conference，去年在盐湖城，今年在达拉斯。两次会议的规模都接近2000人，与会者来自世界各地，不少人已经不是第一次参加。在为期5天的会议上，我每天都能碰到软件开发领域的一些大牛，例如软件工艺领袖Bob大叔（Robert Martin）、极限编程大师Ron Jeffries、精益开发领袖Mary & Tom Poppendieck夫妇等，不胜枚举。

这个已经有10年历史的社区非盈利会议，一直以来保持着“来自社区、服务社区”的宗旨，所有项目均由社区志愿者自行发起并组织，有限的几位全职“员工”仅是作为协调员的角色出现。如果将大会视为一个产品，它的设计师不是任何一个权威机构，而是被称为Stage Producers和Purple Shirts的社区志愿者，他们人数超过一百，来自世界各地，很多人未曾谋面，通过邮件组的方式结识和协作。这种分布式的团队管理，通过可视化、自管理的方式，实现着团队的高效协作。而他们对产品的设计和部署实施，则以部落传染的方式，吸纳众多“用户”参与，甚至是全程参与。这些都能保证大会的设计者敏锐捕捉和满足用户的需求，从而交付一个User-Driven Designed Conference（用户驱动设计的会议产品）。而当一个产品由它未来的用户以满足自我需求的方式、当做自己的作品一样设计时，还担心它不能深入人心吗?



Open Jam开放式自组织主题会议，“会场”设在各个会议室之间的走廊，很多有意思的话题如Agile for Kids、Lean Coffee、Coaching Clinic在这里发生

参加过很多科技会议的人都有体会，与会者排排坐，台上讲，台下玩手机。而这个会议多年以来，都一直保持着很多培训、工作坊、游戏等用户交互很多的环节，还有Open Jam开放式自组织主题会议，No Bulls Know无主题会议。在200多个课堂里穿梭，参与丰富的晚场活动，你总能找到乐子。更不用说还有丰盛的三餐和饮料茶点，吃不完的德州烤牛肉。5天的会议，如果不是摆渡在会场和酒店之间，我几乎没有去过达拉斯的其他地方。

而会议所覆盖的主题则是领先的、全面的。20多个主题会场，议题内容涉及敏捷思想领域的方方面面：组织转型、教练指导、团队协作、领导力、工程实践、敏捷与UX用户交互、敏捷与产品开发、敏捷与创业、敏捷在新兴领域如移动开发中的应用等。而近300个议题，全部海选自全世界敏捷社区的人们提交的1000多个议题。

本次大会的3个Keynote演讲者很有特色，但我更惊喜于这3个演讲者不再是敏捷领域的大牛。按照常规，第一个Opening Keynote被安排在了大会第二天上午（第一天全天是Workshop），主讲者是来自斯坦福大学的管理学教授Robert Sutton，其演讲主题为企业规模化发展的思想、方法和原则。他相信持续的绩效来源于有好的想法、并实施这些想法，他列举了很多硅谷名企的发展历史和研究分析。另一个主题演讲来自印度医疗领域的女性企业家Sunita Maheshwari，她讲述了一位医生白手起家创业的故事，成功的因素在于“持续创新”和“坚持梦想”。

大会的压轴演讲来自一个名不见经传的年轻人，





Joe Justice，他是来自西雅图的敏捷教练，也是WIKISPEED的创始人。他在演讲中讲述了如何用3个月的时间带着团队建造一辆时速超过100英里/小时的赛车，而如果在传统制造车间，这个过程需要数年。

Joe的团队始建于2010年，当时他看到了一个制造赛车的竞赛项目。为了组建团队，他通过不断更新博客和社会化营销渠道吸引人们关注，很快就集结了来自4个国家的44名志愿者。经过讨论，他们向组委会递交了产品原型。在没有多少汽车制造经验，也没有任何资金支持下，他们却战胜了100多个有资金、有经验、有背景的团队。

他们之所以胜出，是因为他们采用了现代软件开发团队使用的敏捷、精益、Scrum等极限项目管理方法（Radical Management），这些方法使他们比传统团队更富有创新性、更迅速。而传统开发方法之所以慢是因为改变的代价太大。在汽车制造业，如果工程师想重新设计车型，那么他需要至少投入100万美元开发模型，然后等待其兑现其市场价值，而这个时间差不多需要10年。会有多少工程师愿意为一个开发周期为10年的产品而苦等呢?

Joe的团队采用敏捷方法，7天一个迭代，极大加速了产品开发的进程。让敏捷不仅发生在了软件开发领域，还运用在汽车制作领域。他认为提升速率的关键在于模块化。而测试驱动开发的方法，又让Joe的团队能够在各个模块开发前做安全方面的设计测试。他们敏捷地管理变更，减少和消除了变更成本，使得变更需求不再成为要等待数年才能实现的难题。

Joe和他的团队的成功还在于，他使用Scrum的方式管理一个分布式协作团队。产品开发不仅仅



Joe Justice已在TED等会议上演讲，并受到福布斯等媒体的专访

是产品经理（或者PO）、开发工程师、测试工程师、设计师的事情，团队所有成员都始终保持着旺盛的士气。团队原有“中层管理者”不再做绩效考核，而加入到了产品设计讨论中，传统的财务、人事等支持性部门的功能被弱化，所有人都投入到产品设计和开发中，他们的创新得到了极大程度的发挥。对于这个成员分布在世界各地的跨国团队来说，团队协作所使用的所有工具全部都是免费的，如FreeConferenceCall、Dropbox、Google Docs、YouTube、Skydrive、Facebook和LinkedIn，所有生产这些产品的企业，创建都不足10年。

Joe和大会中很多演讲者成功的故事还在延续，我们也看到了来自各行各业应用敏捷精神的成功案例。而在国内，“敏捷改变世界”这样的口号，或许尚只在软件行业有限传播。一些只知道埋头做产品开发的人们，或许会感慨敏捷和自己有什么关系——只要开发产品就好了；一些碰巧抬头看到这个名词的人，或许会投机取巧般地给自己贴上标签——只要拿它赚钱就好了；但还有一些默默实践着敏捷精神和方法的人，真正感受到了敏捷带来的改变，并受益匪浅。敏捷的变革性力量，将不仅仅影响着软件开发和生产制造，它将带给世界更大的改变。

跨平台数据库调优利器 DB Optimizer

文 / 周小辉

过去，数据库调优常由有经验的DBA或者外聘数据库专家来担任，代价不菲。能否找到一个性价比高的模式，让普通DBA或者开发人员自己就能胜任数据库调优工作；当需要工作在不同数据库平台时，之前的调优经验依然有用，并且可以防止把性能差的SQL语句带入到生产环节，避免由此带来的返工等诸多问题，成为管理层关心的问题。DB Optimizer（简称DBO）正是这样一款工具：跨平台、运行快速且易于上手、学习门槛低。

DBO是美国英巴卡迪诺公司的7×24小时快速数据库性能调优工具，是一款数据库性能数据采集、分析以及优化SQL语句的集成环境，可以帮助DBA以及开发人员快速发现、诊断和优化执行效率差的SQL语句。**DBO具有中文版**，支持Oracle 、Sybase、IBM DB2、MS SQL Server等数据库平台。

DBO为绿色免安装软件，在数据库服务器端无须安装代理，通过JDBC驱动连接到数据库，无须安装数据库客户端驱动，连接用户只需有权访问动态性能视图即可。DBO在运行时仅采集与性能相关的数据，给数据库服务器带来的系统压力小于1%，因而适合用于关键的生产系统。

DBO的典型工作流程包括：发现和分析问题、解决问题、验证结果。此外，还包括一个用来编写SQL语句的SQL编辑器。

发现和分析问题

DBO包含Profiler、Tuner、SQL Editor和Load Editor四大组件。

通过Profiler组件可以发现和分析问题，判断数据库是否存在瓶颈以及瓶颈的具体所在。Profiler组件持续地对数据库进行数据采集，以构建数据库的负载统计模型。采集数据时会过滤掉执行性能良好的SQL语句，仅收集“重量级”的SQL语句信息【注：“重量级”包括两类，一类是运行时间较长的，另一类是运行频度很高的】。

采集得来的数据可实时显示为数据库负载、Top Activity和Details三部分，如图1所示。图1的顶端区域是负载图，显示了数据库的整体负载，可以直观获知数据库的运转情况。X轴表示时间；Y轴表示平均活动会话数AAS，显示了数据库的性能，当与Max CPU线（图1红线）对比时，AAS可帮助快速而轻松地显示数据库的性能瓶颈：Max CPU线是衡量数据库性能的一个标杆，当

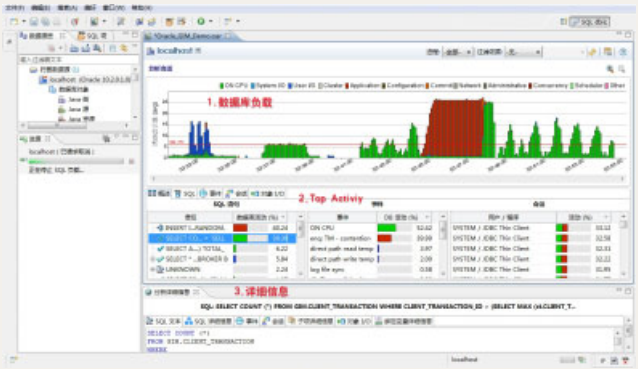


图1　通过Profiler组件实时采集和分析数据库负载数据

AAS超过Max CPU线时，就说明数据库存在瓶颈。

Profiling可以帮助定位和诊断有问题的SQL代码以及基于事件的瓶颈；还可帮助研究分析单个具体的预存程序（包括存储过程和函数等）的执行和等待时间、事件的详细情况，这样就能找到有问题的区域，并进一步定位到具体有问题的SQL语句。

负载图是分析Profiling结果的高级入口点。下面可以使用Top Activity和Profiling Detail视图来查看等待和执行会话的详细信息。

分析Top Activity

Top Activity位于图1中部，用来说明负载的来源，可以进一步细分为Top SQL语句、数据库花费时间最多的Top事件和Top活动会话。这些信息以标签页的形式来组织，每一类放在一个标签页中。还有一个Overview标签页对这些信息提供总体概览。

DBO为SQL语句和语句组件提供了性能统计信息，这些信息可帮助我们评估语句的执行成本以及从中发现存在等待事件问题的SQL语句。对每个SQL语句，DBO提供了所选时间段内的执行次数、平均花费时间、消耗数据库资源的比率等统计信息。

Top Events标签页显示了在Profiling期间发生在各类资源上的等待事件信息，有助于在应用程序和数据库配置层面进行优化。

TOP SESSIONS标签页提供了关于每个会话的信息，可以从中看到那些非常活跃或存在瓶颈的会话的信息。

TOP OBJECT I/O标签页（ORACLE特有）可以显示具体是在哪

个对象上发生了与I/O负载相关的SQL语句、事件和会话。

分析Profiling Details

Profiling Details视图显示了Top Activity视图中所选条目的SQL语句、事件、会话等详细信息，并可从中进一步发现性能问题。

DBO以最优方式来组织收集的性能数据，这使得结果文件非常小【注：具体的结果文件与数据库服务器上事务的密集程度有关】。

性能数据可以文件形式保存，供随后脱机分析，也可保存到专用的性能结果数据数据库中。会话数据可保存的能力非常具有业务价值，例如可以把会话数据文件通过邮件发送给资深专家，该专家无须连接到数据库，将会话数据文件导入到DBO中，从而实现远程“诊断”；也可由DBA和开发人员在**离线**的环境下，在DBO中打开会话数据文件协同解决数据库存在的性能问题。

数据采集之后就是要分析了。如果信息量较大，DBO提供多种手段聚焦数据，包括放大/缩小、过滤、截取一段时间等。可以任选一个时间区段，Top Activity中的数据也会作相应调整仅显示所选区段中的信息。当在Top Activity中选择某个数据时，对应的Profile Detail会显示更详细的信息。

除上述分析方式外，DBO还可以任意的点作为分析的发起点。例如当在Top Activity中选择某个等待事件时，在详细区域将显示对应的SQL语句，还可以双击详细区域中的SQL语句，这时就变成了查看与该语句相关的会话、等待事件、Object I/O等。

在实时关联分析时，如果发现某些负载压力比较高的SQL语句，但由于当时正在处理业务，负载比较高，那么可以将发现的SQL语句拷贝粘贴出到一个SQL文件中，并通过DBO提供的SQL Project统一管理，等到负载不忙时再集中优化。

解决问题（优化SQL语句）

通过Profiling组件发现需要优化的SQL语句后，就可以选择该SQL语句，从弹出菜单中选择“Tune”，就将把该SQL语句导入到DBO的另一组件“调优器（Tunner）”中，从而开始调优。

为了确保DBO的SQL优化器能正确工作，首先必须保证数据库中对象的统计信息是有效的。过时或不正确的统计信息将使得优化器得出错误的结果。

必须说明的是：DBO不仅可以优化SQL语句，还可判断整个应用系统的性能瓶颈到底是在数据库端还是在应用程序端、数据库的配置上是否存在问题等。

DBO不仅可以从Profiling组件中获得要优化的SQL语句，还可直接撰写SQL语句，或从数据源浏览器中拖拽要优化的数据库对象；或选择SQL文件执行批量优化，或从Oracle的SGA中查找要优化的SQL语句。

优化手段之Hints和SQL重写

用户可通过Hints告知数据库优化器执行SQL语句的最优方式。DBO的优化器可以自动地使用Hints生成case，从而得出最优的执行方式。并且，DBO的Hints是可配置的。

SQL重写可以在不改变SQL语句语义的情况下，将其修改成语义上等价、运行效率更高的形式。

索引分析

索引分析功能可以分析SQL语句中的索引使用情况，并判断是否存在应该建而尚未创建的索引，将创建一个虚拟索引，并可应用该索引到数据库中。DBO通过颜色来区分索引的使用现状。

优化手段之VST（Visual SQL Tuning）

VST可以帮助快速找到最优的执行路径。同时，VST可显示一个SQL语句中表与表的关联关系、每个表上的数据行数，当在表上应用了过滤条件后的过滤比和两个表关联时的结果集大小。

VST还可发现SQL语句的缺陷，包括笛卡尔连接和隐性的笛卡尔连接。除了支持普通的SQL语句外，VST还支持外连接和子查询，并可按需展开子查询。

除了上述优化手段之外，还可显示表统计信息、执行列统计信息和柱状图分析、大纲和执行计划。

验证结果（SQL负载压力测试）

DBO提供了SQL负载压力测试组件，以验证经过优化后的SQL语句在大并发、大量执行情况下的负载表现。可以在其中设置压力测试时的并发数、执行停止条件，并设定每次执行的间隔时间。测试语句既可以是具体的SQL语句，也可以是某个SQL文件。可以结合使用SQL压力测试和剖析功能，在加压的同时使用Profiler组件监控数据库的性能反馈，从而获知是否存在性能瓶颈。

结语

本文只是简要介绍了英巴卡迪诺公司的跨平台快速数据库调优产品DB Optimizer，除此之外，DBO还内置了SQL编辑器，该编辑器提供了各种代码助手，可完成代码抽取、语法高亮、自动错误检查、代码完成、代码格式化等功能。限于篇幅，就先介绍到这里。有兴趣的读者，可访问：http://www.embarcadero.com/cn/products/db-optimizer-xe获得更多信息。

“随手记”：借力百度云开发者平台获取资源和信心

记者 / 亦然



“随手记”创始人谷风畅谈百度云平台

“对于每个应用开发者来说，借势平台资源的整合优势，无疑是如虎添翼。百度云开发者平台在随手记的快速成长和品牌知名度的提升方面，起到了重要的助推作用”。面对《程序员》记者的采访，“随手记”创始人谷风向记者回忆了“随手记”的开发历程。

据了解，通过搜索关键词匹配，百度为“随手记”带来了巨大流量。而以随手记为代表的个人理财软件应用，也通过不懈努力获得用户认可。随手记在苹果App Store理财分类榜上连续750天排名第一，被苹果喻为“四大金刚”应用之一的“生产力”应用软件。

抓住移动浪潮 关注用户需求

谷风在金蝶工作了十余年，从2007年起他就对个人财务市场开始了认真的思考。谷风认为，随着个人财务结构越来越复杂，简单的流水账已不能满足很多人的理财需求，需要更方便的理财方式。在他看来，桌面产品向移动互联网切入时，“移动先行”的思想是非常重要的，移动应用的用户增长速度远超桌面端，不能将移动端当成PC端的附庸。

所以在2009年新一代智能手机浪潮来临时，谷风抓住了机会，成立了随手网科技有限公司。多年的财务产品经验累积，让随手记这款产品在一开始就快速地切入个人理财软件领域，加之这个市场在国内的竞争并不激烈，没有太强大的对手，这样的先发优势帮助随手记一路奋力前行。

谷风对记者说道，对于大多数产品来说，其运营数据作为产品方向调整的标杆显得尤为重要。但作为一款涉及到财务信息的产品，其用户数据就变得十分敏感。“由于财务业务涉及用户的隐私，我们基本不对用户数据做任何分析。主要的运营数据都来自各大市场提供的下载和销售数据。”

谷风在采访中明确表明了“随手记”的态度，“用户的数据属于用户，放在任何地方都是用户自己的权利和自由。所以，‘随手记’尽可能地提供更多的方式满足用户需求。”随手记不仅支持云同步功能，同时还支持百度网盘等多种第三方知名品牌云平台。

据了解，在“盈利难”的移动互联网，“随手记”采用了两种方式来获得收入：一是常见的付费下载，二是与金融机构的合作，即随手记把软件付

费授权给银行机构，作为他们为用户提供的增值功能。谷风透露，目前随手记已和银联达成了合作，以后还会与基金公司、银行等金融行业企业一起进行推广合作。

谷风认为，在今年的百度世界上，百度也为开发者推出了一系列应用变现的商业模式，其中除了提供百度移动搜索变现、百度应用与网站变现等广告模式之外，百度已在探索前向用户付费模式。而随手记也将与百度保持更加密切的沟通合作，把握多渠道的变现机会。

在用户付费方面，谷风自信地表示："个人财务数据非常隐私而且重要，用户犯不上为了省小钱而冒大风险。百度的数据平台安全稳定，值得信赖。'随手记'在保护用户隐私方面下足了功夫，由此为自己带来了巨大的优势。"

百度搜索助开发者突破应用商店"马太效应"

一直以来，流量都是每个应用开发者最希望得到的宝贵资源，这也正是百度的优势。通过搜索关键词匹配，例如搜索"记账"、"随手记"就可以直接调用"随手记"App，在为产品带来了巨大流量的同时，也借助百度应用开放平台，为广大用户提供了"即搜即得，即搜即用"的良好体验。

在谈到在今年百度世界大会上，百度推出的"七种武器"时，谷风说道："现在行业说云的人多，做云的人少。虚无缥缈的云多，能够真正'落地'的云少，百度七种武器是百度云能力对外界的一次无私分享。"

谷风透露说："用户使用体验的提升，离不开百度 PCS的支持，它为开发者节省了大量的存储服务器的购买和运营成本，节省的费用至少是百万级的，这对于许多创业公司来说，减少不小的运营成本压力。"

对于BAE（百度应用引擎），谷风也比较推崇。他介绍说，借助 BAE 基础架构，开发者可以不需要维护任何服务器，只需要简单地上传应用程序，就可以为用户提供优质服务。同时，百度MTC（百度移动云测试中心）也解决了多终端的测试问题。据悉，通过百度 MTC，开发者可以利用该中心提供的几十种主流厂商的移动终端设备及模拟器，并进行实时的手机应用开发和测试工作，包括 UI 适配、稳定性测试等多种测试类型。

在采访中，谷风还介绍道："百度云平台的支持，可以让开发者减少管理外围技术、带宽、服务器等问题的负担，继续专注于业务逻辑、产品创新，为用户带去更好的理财体验。"

百度云与开发者共赢

2012 年，各大巨头纷纷强势进驻移动互联网。在通用市场上，巨头们占据着资金、渠道等巨大优势。在谷风看来，创业型企业成功的唯一招式就是——"做巨头不愿意做的市场，并跟随细分市场一起长大。"但这类市场的规模往往太小或太深，看上去"不起眼"或做起来"太费力"。谷风认为："能否看到市场无疑需要考验创业者的功力，而与巨头的合作能够带来更多的资源和支持，解决'太费力'的难题。有了金蝶与百度长期的良好合作，也使得"随手记"在一开始就选择加入了百度云开发者平台。"

与百度合作后，随手记获得了的很全面支持，在"开发、运营、推广、变现"等各个方面都有所收获。目前，随手记Web应用在百度记账理财榜上排名第一，使用次数已超过450万，超过后八名应用产品使用次数总和的两倍多，成为了当之无愧的明星记账应用产品。目前，随手记已得到超过2000万用户的认同，VIP用户也已突破了50万。

在日前刚刚结束的"百度云应用开发大赛"中，随手记从5000多个参赛应用中脱颖而出，获得"最受欢迎应用银奖"。谈到未来谷风表示，随手记将继续在个人财务领域进行深耕，同时加大行业间的合作。

"个人财务将不只是个人记账这一小范畴，随手记的信用卡管理应用'卡牛'在不到半年内已发展了超过500万用户，说明这个市场的巨大潜力"，谷风认为："从总体来看，还有非常多细分领域等待我们进入。同时，现有的产品还不能完全的满足用户体验需求，这也是我们继续努力的方向。"



移动互联网
创业·求生

移动互联网蕴含着巨大财富，但与此同时，创业之路却举步维艰：产品同质化竞争加剧，推广成本不断攀升，盈利模式迟迟未能清晰，投资者忽冷忽热……究竟要跨越多少道坎，我们才能生存下去？

本期封面报道，就移动互联网创业路上的关键“痛”点，我们邀请了前沿观察者、投资者、渠道商、数据平台以及亲身实践者们，分享自己的思考与心得，帮助更多创业者们在求生的十字路口做出更好的选择！

无论如何，“活下去，才有希望”。



2012，当我们谈论移动互联网创业时，我们在谈论些什么？

文 / 陈粲然

2012年的移动互联网市场可以用“冰火两重天”来形容。

根据Google官方数据，Android设备激活量在2012年3-6月4个月内，由3亿部增长至4亿部，增长率为33%。据苹果官方数字，iOS设备激活量从2012年4-6月由3.6亿部增长至4.1亿部，增长了12%。而对于中国市场，根据友盟的数据，2012年第二季度Android App启动次数增长159%；iOS活跃设备增长42%，iOS App启动次数增长112%。这些数字都表明了移动互联网用户和终端量在2012年仍在高速增长。

但另一方面，移动互联网创业之路却愈发举步维艰：同质化竞争加剧，推广成本攀升，盈利模式迟迟未能清晰，投资者的钱也烧得差不多了……

但移动互联网始终是座巨大的宝藏，吸引着无数人甘愿冒险探寻。那么当前，如何度过寒冬，成功地求生是每个创业者、投资人都在思考的问题。本文中我将列举几个2012年移动互联网创业的关键点进行讨论，内容仅供引发思考，不做结论。

创业机会在哪里？

洗牌前期，巨头们开始以各种方式大肆进驻。一方面是以腾讯为代表的巨头正在进行全面的软件布局。目前腾讯已推出了几十款覆盖各个平台的App，涵盖了社交、浏览、订阅、输入、同步、安全、音乐、游戏、阅读等领域，以强大的资金、渠道和人才优势与大批创业者直接竞争。另一方面，百度、阿里巴巴、奇虎360、盛大等公司开始竞相推出搭载自己定制的操作系统的手机，抢占移动互联网的入口……

如果比拼渠道、资金和对人才的吸引，小公司毫好无胜算。因此，在巨头驻足的通用领域，采取与巨头直接竞争的方式异常艰难。那么，在这种环境下创业者在创新方向选择时可能需要考虑“垂直化”和“行业化”，更多地去挖掘那些还未被发现或者巨头没有兴趣去做的细分市场。在游戏、SoLoMo、O2O、移动电商、儿童教育等直接可以从消费者手上收钱的领域都有机会，在整个行业的真实收入迅速攀升的基础上，移动广告、开发工具、各种外包等周边产业也会迎来相应的繁荣，与之相关的风险投资、孵化器、财务、法务、媒体、人力资源等领域也会被拉动，各种企业的移动化也需要“行业化”应用的推动。在这些方向上，都尚未出现真正集大成、占据绝对优势的产品，如果能做出好产品加上合适的

分发渠道，创业者仍然有机会。

同时“硬件化”也是一个选择方向。随着中国二三线城市智能手机的普及，智能手机的细分市场也开始成型，针对不同性别、层级、年龄段消费者的手机需要配套的软件和系统定制，因此这个层面上创业者与厂商间的软硬整合合作也存在一定的机会，当然，其中还包括平板、智能电视等领域。

Native App还是Web App?

记得2011年时，Web App的概念大行其道，同时伴随着Adobe Flash Player的诸多不利消息，一时间HTML5被一些人捧上了制高点。但近期，Flash产品（首当其冲是游戏）开始全面爆发。依靠着Adobe AIR、Stage 3D等强大技术的布局，移动设备上开始涌现大量高品质的Flash游戏。而HTML5却从云端跌回残酷的现实：HTML5移动游戏领域的早期探索者Moblyng倒闭，Mark Zuckerburg亲口承认Facebook的HTML5战略是个错误，Chrome Web Store里面力推的游戏表现都不甚理想……由于表现能力不足，自身标准一直未确定，并且缺少Web App分发和盈利的渠道，导致HTML5阵营中并未出现拿得出手的成功案例。也许就像一位开发者曾经说过那样：“HTML5是未来的趋势，但不是现在的优势。”

毫无疑问，每种技术方案都各自有其优劣之处，必须要根据实际情况灵活选择。例如对于许多交互不多的传统内容型网站来说，HTML5已基本可以覆盖全部需求。但在目前Web App受限于浏览器或前端技术未标准化的情况下，开发者可以考虑进行组合应用，对于侧重性能、体验、设备特性、本地数据管理等HTML5表现力达不到的部分用Native来做，HTML5表现力能达到的部分就用HTML5来做。

总的来说，开发者应该掌握并提前布局多种技术，不能在一棵树上吊死。

最好的盈利模式?

无论怎样，成功不是一时半会儿的事，“活下去”是当前最重要的话题。由于投资锐减，创业者需要更多地、更早地去考虑盈利的问题。其实目前的移动互联网有一系列的盈利模式可供开发人员选择：

- Pay-Per-Download。下载支付，为获得App使用权一次性支付费用。
- In-App Purchasing。应用内购买，通常是一些应用的功能、提升游戏等级以及虚拟商品，目前是大陆最容易获得高收入的模式。Subscriptions。订阅，一般是逐月为App付费，《程序员》杂志等阅读类App通常会采取这一模式。
- Freemium。免费下载，提供增值付费服务，在诸如Evernote等工具类应用中常见。
- Commissioned Applications。提供外包服务。
- In-App Advertising。App内展示广告，这一种广告模式和传统互联网的广告模式十分接近，被开发者大量使用，但效果渐减。
- Product Placement。植入式广告，例如在游戏中的某一道具就是某品牌的饮料。
- Purchase Intermediation。为从第三方App中带来的流量或其他资源拿到分成，但在大陆也常常演变为资源、流量互换。
- Per-unit royalties。通常是与终端厂商和各大平台合作，例如厂商为手机上预装付费等。
- Distribution Exclusivity Deals。针对独家应用商店发部应用，前段时间Rovio的《Amazing Alex》通过应用汇发布时似乎就采用了这种模式。

在2012年6月VisionMobile对全球1500名创业者的调查中发现，Pay-Per-Download是当前最频繁使用的盈利模式。共有34%的创业者选择了这种方式，较2011年相比略有下降。紧随其后的就是In-App Advertising，占33%。而从今年起，In-App Purchasing模式开始火热，已跻身高回报的收益模式之一。

但创业者需要明白，没有最好的模式，只有最适合的模式。而选择标准往往取决于他们的产品模型、规模和目标市场。例如在大陆，游戏用户很难为一款未知的游戏下载付费，因此In-App

Purchasing在今年大肆流行。而对于一些理财类工具，用户反而愿意付费来选择高品质的产品来保证自己的安全和隐私。同时，对于同一款产品，在不同阶段“盈利模式”也可以转型，《Angry Birds》就是最好的例子，在长期霸占App收费榜前排位置之后，随着新产品的推出和游戏生命周期到来，小鸟们已从排行榜上跌落，却找到了更为稳健的盈利模式：其周边产品和授权费为其带来了丰厚的回报，当然在大陆这种模式可能会被山寨得让人吐血。

不过在2012年9月17日，App Store出现7小时的“锁榜”现象（指整个榜单全部锁住，排名几乎不动，偶尔有一两名上下的波动只是因为某些限免App到期，从免费应用榜单下架所致）。这个情况与2012年3月20日（上一轮排名算法完全改变的前2天）出现的7小时锁榜现象如出一辙。因此，有人预测排名算法即将再次大调整，希望会带来一些积极有效的变革。

生态系统困境

再好的产品也需要依靠良好的分发渠道和健全的市场，可2012年的中国移动互联网生态系统可谓满目疮痍，大部分开发者都多多少少面对迷局。对于中国开发者而言，山寨术依然是亘古不变的话题，而在这之外，新的问题接踵而来。

App Store一度帮助苹果打造了最优秀的移动生态系统。但2012年的种种迹象表明，这片生态系统似乎正在遭受侵蚀。刷榜问题日益严重，在2012年3月时，App Store曾经采用技术更新的方式阻止刷榜，但也默认了对之前刷榜严重的公司不追究。果不其然，苹果的这种“不作为”态度让新算法仅见效2个多月，一股新的刷榜风潮就开始愈演愈烈。有开发者认为苹果应该像Google Play那样摆出“令可错杀一千，不可放过一个”的态度，然假设如此，想必肯定会有无良对手花钱帮你“刷榜”，让你欲哭无泪。

同样，在Android的第三方市场，刷榜也是同样泛滥。因此，有许多国内的开发者和媒体开始曝光和谴责这些产品。但从市场层面来看，这种影响并不会波及到消费者，倘若刷榜风险和成本极小，加上一些投资人“有钱赚，怎么赚都是赚”的纵容态度，那么必定会有更多的人铤而走险。

除此之外，App Store审核周期常常是漫长且毫无规律可循；去对手产品下面刷差评；有程序员直接拿别人游戏安装包破解反编译加入自己的广告账号，放渠道上偷广告流量……开发者不仅要注意暴风雨和暗礁，还需要迎战其他船只的攻击，移动互联网生态圈的困局还有多久才能解开？

走向海外？

虽然有一些应用在中国也取得了不错的成绩，但暂时而言，大陆终归是付费意愿较低的市场，并且还有着前文提到的各种恶劣的竞争环境，在此状况下，走“国际化”战略似乎是个不错的选择。实际上，如今中国移动互联网的很多明星团队已走向海外、专注海外甚至如《海豚浏览器》这样“来自海外”。

成功者固然令人艳羡，但成功的路却并非所见即所得。作为一个统称，海外市场实际上被细分成很多个不同的市场：欧美市场、日本市场、韩国市场、拉美市场、东南亚市场……每个市场都依靠语言、文化、渠道特色、市场机制等地域特点建立了壁垒，倘若没有找对 法门，可能不仅没有找到宝藏，反而撞得头破血流。简单地以日本市场为例，虽然其ARPU值非常高，但日本市场的用户口味与其他国家有很大区别，盈利主要来自卡牌游戏，而且日本用户对于美术画风、各种细节的要求非常高（例如相比其他亚洲市场，欧美风格的产品可能更受欢迎）。

另外，也不是所有的产品都适合走向海外。很多产品一开始就拥有中国市场的基因，离开这个环节与其他本土产品竞争，很可能水土不服。在近期我采访的几位开发者中，他们在谈到海外市场时纷纷谈到中国团队的技术优势，看来这一点是每个想要异国求生的团队必须具备的素质。

移动互联网创业路向何方？

文 / 庄明浩

相信大家比较有共识的一件事是，中国是从2010年起进入真正意义上的“移动互联网”时代。那一年我们彷徨、迷茫、兴奋，各式各样的“纯”移动互联网项目兴起，这一波创业热潮在2011年达到巅峰。行业外的人笑称，在2011年，拉几个兄弟，做一个App就能得到融资。但这一波浪潮又迅速地进入所谓的“波谷”。2012已过去了2/3，相对于2011年的“整体疯狂”，我想无论是创业者，还是投资人，抑或是第三方的观察者们，对于移动互联网创业的大环境都有了比较清晰的认识：终端爆发、模式多样、资本趋冷、盈利压力……而从我个人的角度看，就像灾难电影《2012》所展示的那样，在这场“浩劫”之中，我们要看重：“钱、前和潜”。

2012要向钱看、向前看、向潜看

向“钱”看

或许在2012年，所有移动互联网创业项目被问得最多的问题已不再是“腾讯做了怎么办？”，而是“你靠什么挣钱？”。在资本趋冷的大环境下，找到“钱”对于我们这些“互联网/移动互联网人”太重要了。十几年前互联网在国内刚刚兴起时，我们可以说：“用户数为主、流量为主，商业模式都是摸索出来的。”但在2012年，移动互联网的早期项目可能要更多地承担起来自“收入”的压力。在行业兴起之初，我们或许可以用例如“刚需”、“入口”这样的说辞解释项目初期的无收入，但经历了2年的发展，这个说辞可能已不完全适用。

向“前”看

“前”是“前进”的“前”，也是“前夜”的“前”。2012年不好过，是几乎所有创业者和投资人的看法，但同时大家也都意识到了行业的未来一定是美好的，目前的“苦逼”或许只是阶段性的。因此对于创业者来说，今年最重要的是事情是做好准备，可能包括但不限于如业务流程、产品迭代、团队打磨等方面。因为无论是“好”时光，还是“坏”年代，这些才是你的核心竞争力。

向“潜”看

“潜”是给那些在2011年获得了一定资金支持的创业者的建议。首先要恭喜那些在资本的高峰期以高估值获得了高于行业平均水平规模融资的“三高”企业。但同时2012年对于它们也是十分艰难的：因为你的“三高”投资者对于这些创业项目的要求会更加严苛，而项目本身如果不在2012年利用自己在资本上的优势“潜身”修炼自己的内功，增强自身的实力，那么到了下一轮时，或许等待你的只有“Down Round”（“估值打折”，指投

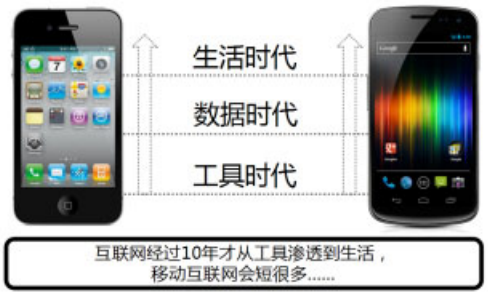
资者在一轮融资中购买同一家公司股票的价格低于对上一次融资投资者支付价格的情况）。

移动互联网发展的趋势逻辑

中国移动互联网发展到今天，创业者选择创业方向时要认清行业的大趋势，同时也要知道这些趋势后的大逻辑。从一些业界人士的角度看，2012年的移动互联网发展可以总结成两种逻辑：时代更迭和“屌丝”经济学。

工具时代→数据时代→生活时代

“时代”论的说法我是借鉴了现长城会日本区董事长、前DoCoMo副总裁夏野刚的理论。简单地理解为移动互联网的发展会经历大致三个时代：工具时代、数据时代和生活时代。当然，在每个时代的更迭过程中，上一个时代的业务也会有渗透和变化。



工具时代即以工具类项目为主的时代。一般是在智能机刚刚兴起之时，会出现很多弥补智能设备或者智能网络本身缺陷的项目，这个时代的用户是第一批愿意尝试新鲜事物的所谓Geek人群，他们对于用户体验、功能挖掘有着近乎痴迷的态度。

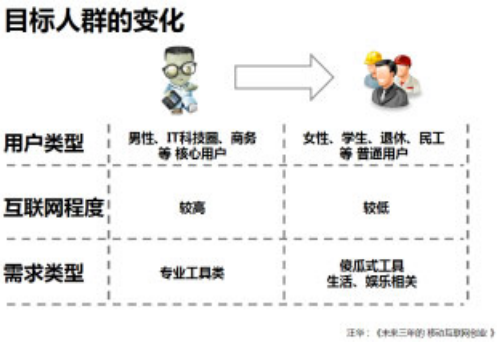
数据时代则更多是从运营商的角度看行业发展，即有很多基于移动网络数据的具体应用出现；而到行业发展的最后，随着智能终端的普及，移动互联网开始渗透入人们生活的方方面面，生活时代最终来临。

用目前最“庸俗”的移动互联网名词来区分各个时代的话，数据时代的标签应该是“游戏”，而生活时代的“标签”就是“O2O”或者“SoLoMo”。与此同时，任何一个时代都会存在提供基础服务的平台级项目。而在国内，由于发展过快，我们正处在一个“工具+数据+生活”的“混合”时代。



“屌丝”经济学

这一观点来自于创新工场管理合伙人汪华。2012年中国将迎来从5000万智能终端（iOS+Android）到亿级终端的时代，终端的普及就意味着人群的覆盖度更广泛，智能机已不再仅仅是Geek人群手中的玩具，移动互联网将从专有人群开始向平民百姓的过渡。人群的变化直接带来了用户使用习惯的巨大变化，进而影响到整个行业的商业模式变化。类似的变化中国的互联网也经历过，最直接的体现就是出现Hao123、360安全软件（对比之前的卡巴斯基、瑞星等专业杀毒、防火墙软件）、美图秀秀（对比专业工具Photoshop）等。新商业模式的兴起完全颠覆了Geek人群对于整个互联网的认识，更傻瓜化、简洁化、直接化的工具/产品获得了更大规模用户的认可；在大量用户的基础上，这些新型的工具/产品开始找寻到了软件付费之外的新的商业模式。



具体的方向选择

大致看清楚了行业的发展趋势，也大体了解了其

背后的逻辑，我想很多人在选择创业方向时可能就会更靠谱一些。

工具类：国内收费没戏，要么“屌丝”化，要么去国外

目前国内的各种工具类目（浏览器、输入法、短信增强、安全、拍照等），基本都已有了所谓的“寡头”垄断之势态。可能TOP的项目中有大公司（腾讯、百度等），也有创业类型公司，对于这些在某一细分工具类目前还在第一梯队坚持的创业团队来说，“钱”是他们最大的劣势。那么在这一年里，这些项目要么选择做海外收费版本，要么开始尝试Social化或者所谓的“屌丝”化。而对于新进入者，除了再细分，我可能想不到更好的方式；但同时要注意的是过分细分后是否有足够的发展空间。



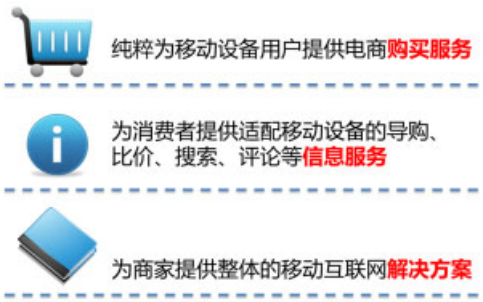
游戏：国内做收入，国外做梦想+收入

公认的，游戏是目前整个移动互联网最挣钱的方式。从目前已在做的移动互联网游戏项目看，2012年过去的9个月有几个比较明确的趋势，例如：页游/Social游戏手游化、传统WAP/KJava转向智能机平台、3D化趋势明显、重度计费产品国内流行。

但同时，游戏行业也面临资本不认可的问题，相信大家也都知道国内网络游戏公司的PE已经低到什么程度（盛大游戏5.56、完美3.66、巨人6.04、畅游4.66），同时Zynga已跌到市值刨除现金仅剩2亿美元的尴尬地步，因此，移动互联网游戏虽然非常热，可真正意义上的资本介入并没有大面积爆发。那么对于选择游戏行业的创业者而言，保持低成本前提下的高收入是唯一的目标。目前相对于成熟的国内端游市场、页游市场和Social游戏市场，手游行业还远远没到可以下定论的年代；但反过来说，在某些细分领域，国内的手游行业已完全成为红海，创业者选择进入这样的细分方向时就要问问自己的团队是否足够强，强到干掉所有其他人。

移动电商：卖东西、卖信息、卖人

移动电商是看上去美好，但实际上是艰难的细分行业。我个人认为，现在移动电商业务的项目可以分成三类：卖东西的、卖信息的和卖人的。

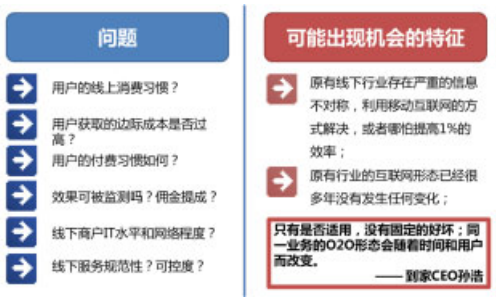


对于“卖东西”的移动电商项目，面对的依然是和互联网电商相同的问题：品类、库存、供应链、成本、毛利等，而目前移动电商的优势可能在于用户获取成本较低；而信息类服务，可能是目前移动电商领域项目最多的细分方向，但这类项目现在更多的是依赖淘宝客，同时在短期内很难获得爆发式的用户增长；而最后一类——为商家提供移动电商解决方案的项目，更类似传统的IT外包服务型公司，这类公司对于投资者的吸引力可能来自公司规模、收入规模和行业地位。

O2O：闭环

因为工作的关系，我最近比较集中地在看O2O领域的创业项目，对项目的评判其实更多地来自一些固有的标准，例如团队构成中是否有懂线下的人、线下行业的互联网化进程、效果的可量化性、佣金比率、潜在市场份额、用户的线上消费习惯、消费频次等。这种标准本身没有什么不对，

但很多时候我们其实是用了一些过分严苛的指标去衡量一个远没有成熟的细分领域。



其实在特定的领域和市场环境下，上面所列举的各种衡量标准的权重应该非常低，甚至可以几乎忽略不计。正如同样在探索线下餐饮O2O业务的到家美食会CEO孙浩所说：“O2O的生意只有是否适用，没有固定的好坏；同一业务的O2O形态会随着时间和用户而改变。”

BaaS（后端即服务：Backend as a Service）：尽量获得收入和盈利

BaaS的概念沿袭自云计算里的PaaS（Platform as a Service），可以理解成是PaaS在移动领域的一个变种。这类公司为移动应用开发者提供整合云后端的边界服务，类似的服务在美国已经非常流行。



在功能上，一般BaaS提供：数据库管理、后台数据存储、地理位置、信息推送、移动App的CRM、统计、社会化分享等。所有的功能一般由调用API或者内嵌SDK的方式实现，开发者只需要关注自己App的核心功能需求，其他功能直接由这些Baas提供标准且傻瓜化的API/SDK。而在商业模式上一般采取月费，或者根据用户量、数据大小、请求次数等收费方式。中国的开发者已经够多，因此，理论上应该有给这部分人群提供服务的项目出现，但国内开发者对安全的顾虑和付费意愿可能在短期内较难有改观。

其他

除了上述细分方向之外，目前国内移动互联网创业还出现了一些新的方式，这些方式的共同点是业务本身来自一个已存在多年的行业（不一定是我们平时所想的所谓传统行业），移动互联网给这些行业带来了一些新的变化，例如法律、金融、医疗和教育等。但这类创业项目一般投资者会对团队有更加严苛的要求。

写在最后

其实上面所列出的移动互联网方向不过是众多项目的一种归类，但因为永远有新的项目出现，所以归类这件事情永远是动态的。同时每个类目都有非常多的内容可以深入写出，本文更多是对于移动互联网行业的基础认识，无关对错，文中有很多观点值得商榷，如果读者有兴趣深入讨论，可以与我联系。



庄明浩
经纬中国分析师，主要关注于无线互联网、社会化媒体和网络安全领域的投资。
微博：@经纬庄明浩

移动互联网创业回归内容本身

文 / 友盟

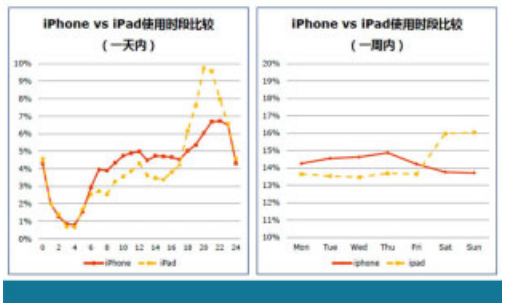
目前，移动互联网从高速野蛮发展期转入相对平稳的时期，用户习惯逐渐稳定，各个分类中都已经存在不少用户忠诚度很高的大牌应用，在这个时期新进入移动互联网领域，创业方向上的选择十分重要。系统、设备、用户行为、竞争对手……所有的创业因素中都暗含风险与挑战，当然也隐藏着巨大的机会。如何发现这些机会？数据将是一把标尺。友盟通过对超过40000款移动应用的数据进行统计挖掘，分析出下面几个比较明显的趋势，或许可以为将要进入移动互联网行业的开发者们提供一些参考。

娱乐类应用值得关注

移动终端正在极大程度地渗入人们的生活，几乎可以说无处不在。近两年来，基于移动终端的娱乐内容不断发展，用户的娱乐习惯逐渐被培养出来，都市生活节奏中碎片化的时间都被移动应用抢占。据友盟对App Store的调查，2012年1-6月，中国市场TOP300中，娱乐类的App数量有所增长，这或许可以对娱乐类App呈涨势这一观点进行佐证。

人们对娱乐类应用的需求越来越大，而开发者也越来越多地进入到这个细分领域中。从注册使用友盟统计产品的开发者来看，近半年来娱乐类App的开发数量有上升势头，7月娱乐类开发量是1月的2.3倍，占到全部应用的14%左右。优秀的娱乐类应用，平均每次使用时长在80秒以上，最高可达300秒以上，高于其他应用的平均水平。

需要注意的是，在开发娱乐类应用时，要注意它们分布的终端。一般来讲，在iOS设备中，相对于iPhone而言，iPad更倾向于满足休闲娱乐的需求，因为它一般在周末和晚间被使用，而这些时段是娱乐类应用使用较为集中的时段。而iPhone使用黄金时段在7：00~17：00，其中阅读、新闻类应用使用频率较高。由此看来，开发者一来可以挖掘iPad上娱乐类应用的机会，二来也可以多多研究平日人们的琐碎时间用来做了什么，把握这些时间和关注点，对开发出优质应用来讲是最关键的。



此外，娱乐与社交的结合是此类应用发展的一个方向，因为人们不再满足于“自娱自乐”，而是希望能通过轻松的方式和有趣的内容，将自己与朋友联系起来。从Jing.fm、你听等音乐社交网站，我们已经可以发现这样的趋势；而在这个趋势中，毫无疑问，精准的内容是最核心的要素，内容导向的产品将引起用户更多的共鸣。

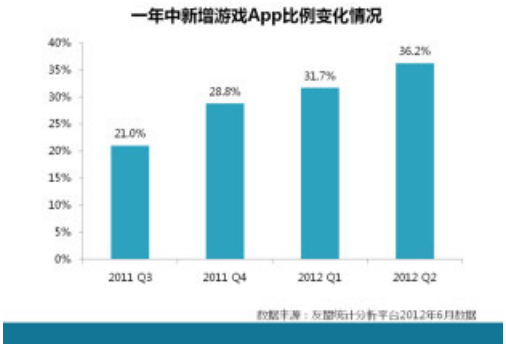
掘金移动游戏

移动游戏的火暴无需多说，看看App Annie的统计数据就知道：2012年7月，App Annie披露的中国iOS收入排名前10位的发行商，全是游戏开发企业。

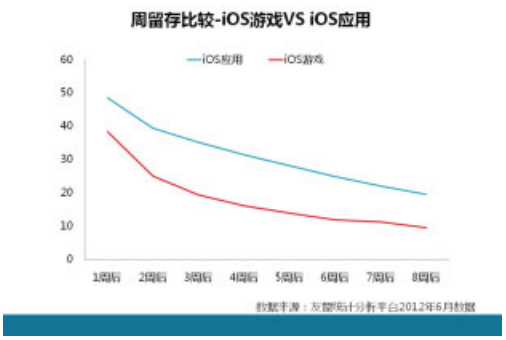
追踪苹果应用商店中的游戏App，我们发现，在Top 300中，国内游戏开发者自主研发的游戏在所有发布的游戏里所占比例稳步上升。在2012年3~6月间，从近35%的比例提升至40%以上，无论是大型公司还是创业团队的表现都令人瞩目。《天朝教育委员会》、《神仙道》、《魔界勇士》等，在核心玩法、游戏性和商业上都取得了成功，是值得称道的原创作品。

整个手游市场的火暴，也吸引了更多的开发者加

入到游戏开发的行列中，从注册使用友盟统计的游戏来看，季度新增App中，游戏占比从2011年Q3的21.0%到达2012年Q2的36.2%，每个季度都有20%~30%的比重，且一直处于上升趋势。



除了游戏的推广，开发者还要关注的一个问题，就是游戏的留存率。从友盟统计分析平台TOP 100游戏和TOP 100应用的数据情况可以看出，游戏的留存率整体上要低于应用的留存率，应用的次日留存可以达到30%，而游戏的次日留存则在25%左右，8日后应用的留存还有将近16%，而游戏的留存不到9%。在周留存方面，第二周是游戏的一个拐点，这一周内游戏留存的下降较多，从38%到25%，而应用的周留存变化则相对平稳。由此可见，游戏开发者们还普遍面临用户周期性流失的风险，特别是单机游戏。游戏内容的设计和管理，成为开发者面对的最为重要的课题。



移动终端平台的游戏一直是友盟集中精力关注的领域。友盟也一直在致力于为游戏开发者提供更好的统计产品。在未来产品、开发者和用户大爆发的时代中，数据挖掘和处理将会更有价值，数据下彰显的逻辑，将带领开发者更好地挖掘游戏领域的金矿。

Android 4.0，越成长越精彩

做什么应用是一方面，做哪个平台的应用是另一方面的问题。在Android系统版本中，2.3占有最广大人民群众基础，备受关注，但4.0版本同样吸引开发者们的“眼球”。2012第二季度数据报告表明，截至2012年6月， Android 4.0版本的设备在总体Android活跃设备中已经占比8.9%，且用户较Android 2.3更为活跃，平均每活跃设备App月启动次数是2.3用户的1.7倍。而2012年7月，Android 4.0设备的比例进一步增长，达到全部Android活跃设备的14.0%。Android 4.0要求厂商和开发者投入越来越多的精力。

Android 4.0的用户有何特征？友盟对统计平台的数据进行分析后发现，目前Android 4.0活跃设备中，数量排在第一位的是华为C8812，也许这出乎很多人的意料。TOP10中有七款出厂即为Android 4.0配置，另外三款则是Android 2.3升级而来。它们是：小米M1、三星Galaxy SII、HTC Sensation XE。

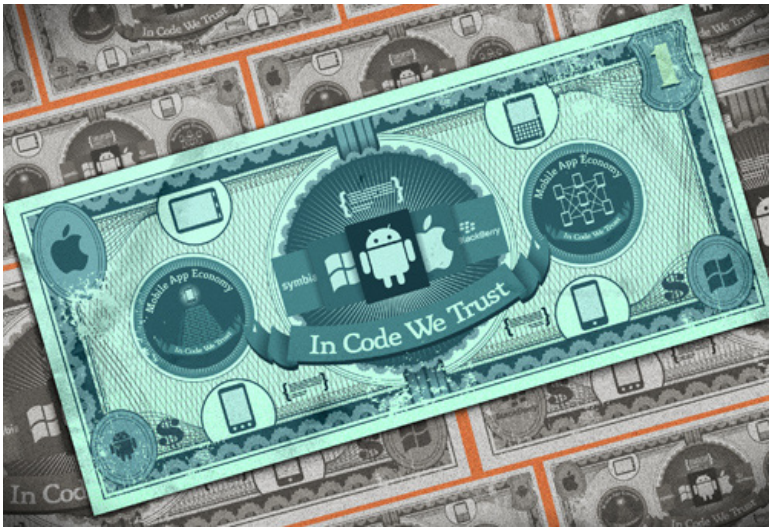


此外，与7月整体数据相比，Android 4.0用户使用Wi-Fi较多，使用2G和3G的比例略低于整体Android用户。这说明4.0用户在使用Wi-Fi的场合停留更多。

由此看来，Android 4.0的用户并不仅是“极客”和“高富帅”。最活跃的4.0设备中，更多的还是一些普通的用户。随着更多优秀的Android 4.0设备出现在市场上，这一类用户将越来越多，到那时，开发者的空间将更为广阔。

2012移动开发者经济学

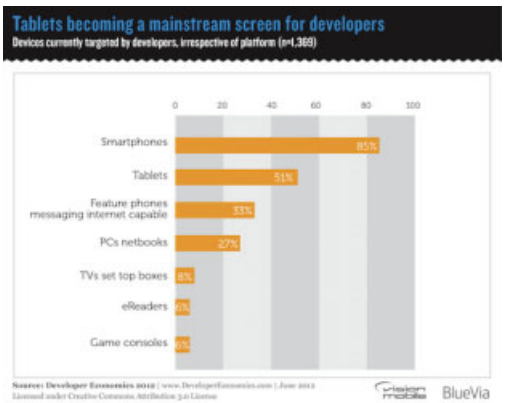
文 / VisionMobile　译 / 陈粲然，宋子齐



VisionMobile每年都会发布移动开发者经济学报告，在《2012年移动开发者经济学报告》中，他们分析了终端厂商的崛起和衰落、移动终端市场分布状况、开发者生存状况和全球应用市场等内容，本文译自其总结报告，并进行了部分删改。报告全文可在http://www.visionmobile.com/devs下载。

新的终端制造商格局。曾经被诺基亚、摩托罗拉、索尼爱立信等公司统领市场的局面早已不复返了。在新的格局中，苹果凭借不断的创新引领着整个市场，三星依靠跟随策略紧随其后，ZTE公司则在组装产业上领先于其他公司，诺基亚虽然在智能机市场一败再败，但仍占据着功能机市场的巨大份额。通过紧密整合的硬件、软件、服务与设计，苹果提供了独特的产品体验并攫取了近 3/4 的行业利润。而三星的的营收位居第二，作为一家快速模仿者，三星的成功秘诀是携手每个Android新版本抢先占领市场，同时由于自行生产芯片组和屏幕（硬件组中最昂贵的两个组件），三星保证了利润和新组件的面市速度。

平板电脑市场愈发重要。开发人员正在快速应对日益普及的平板电脑：根据调查发现，无论何种平台，都有超过50%的开发人员会为平板电脑开发应用，尤其是在iOS平台，有74%的开发者这么做。与2011年只有34.5%的开发人员在为平板电脑开发应用相比，这是一个巨大的增长。与之相比，智能电视和游戏终端就略显惨淡，只有不到10%的开发人员在为它们开发产品。



平板电脑在开发者心中的地位有了巨大提升

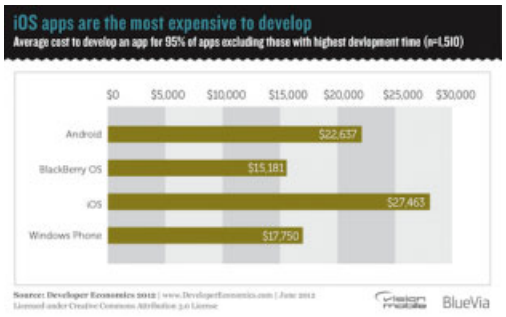
用户数是平台选择的根本。在调查中，54%的开发者认为庞大的活跃用户仍是平台选择的首要标准，另有43%的开发人员选择了低成本。出人意料的是，只有30%的开发者会因收入潜力来选择平台。总的来说，用户数量和赚钱是平台选择的根源，因为一旦App有了活跃用户量，广告、下载、订阅或其他盈利模式就会随之而来。

1/3的开发人员生活在应用程序贫困线以下。基于VisionMobile对1500名开发人员的研究发现，尽

管平均每个应用程序的月收入在1200~3900美元（具体取决于平台）之间，但一个应用程序收益1~500美元的占比为35%。这就意味着1/3的开发者生活在“应用程序贫困线”以下。也就是说，他们无法依靠应用程序作为唯一的收入来源。另外有14%的开发者每月能依靠应用程序获得500~1000美元的收入，有13%的开发者每月获得1000~5000美元。

黑莓是平均收入最高的系统。黑莓开发人员每月每个应用的平均收入为3900美元，比iOS多了4%，而后者的收入比Android开发人员多35%。iOS战胜Android得益于优良的用户特征（苹果用户更少关注价格）、优秀的内容（付费应用程序与免费应用程序的比率更高）、平板电脑的主导地位（每个应用程序的价格更高）和顺利的支付过程（4亿备案的账户只需一次单击即可支付）。

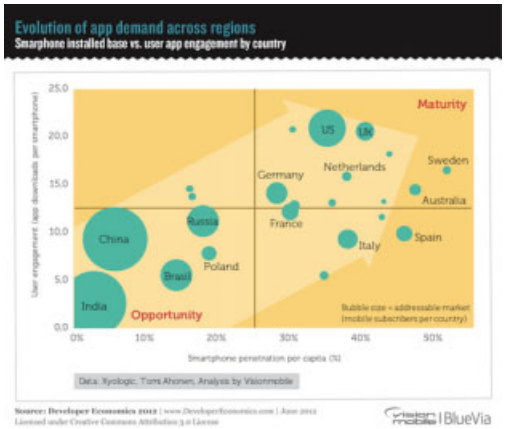
iOS是开发成本最高的开发平台。在iOS上开发应用，平均每个应用程序需要27000美元。比Android高21%，比黑莓高81%。一般一个应用程序需要3个月的开发周期，且应用程序的开发成本取决于具体的地区和应用程序类别。例如在iOS上开发通信和社交网络应用程序比Android周期更短。



iOS的开发成本最高

运营商渠道的边缘化。运营商门户网站的使用率下降了47%，在各个平台上仅被3%的开发者用作主要渠道。但有一个例外：由于Google Play在中国的缺失跛足，中国运营商的应用商店拥有22%的开发者，为消费者提供了超过6亿次下载（数据来源于IHS Screen Digest）。

下1000万个应用程序下载量在哪里。北美市场具有最高的应用需求，对于41%的开发人员来说，这始终是下载量排名前3位的区域。欧洲占了31%的份额，接下来是亚洲。应用需求的增长有3个因素：智能电话的市场占有率上升、用户范围扩大提升，以及一个整个市场的潜力。尽管许多西方市场已经占据了许多这些因素，但BRIC市场（“金砖四国”，指巴西、俄罗斯、印度和中国四国）的市场需求和智能电话的增长数要比西方市场高一个量级。所以接下来的1000万个应用程序下载量应该不会来自当前的领先市场，而是来自BRIC地区的需求。



BRIC市场具有巨大增长潜力

应用程序语言的不平衡。将全球范围使用的语言与开发人员提供的应用程序语言相比，本地语言出现了缺口：85%的应用程序是英文的，但全球英语人口只有8%（约5亿人），而占全球22%人口的中文应用仅吸引了16%的开发人员。欧洲开发者平均使用2.45种语言发布产品，而在南美洲，84%的开发人员使用西班牙语，英语的使用比例仅为48%。因此，以本地语言开发的应用存在巨大缺口。

绘制全球化贸易路线。拉丁美洲拥有最大比例的开发者（44%），他们的应用也是面向全球市场发布的。相比之下，欧洲市场是出口最少的体系，只有26%的开发者专注于非区域性市场。总体来讲，解决自己所在地区需求的开发者比例比解决全球市场需求的开发者比例高得多。这个环境下，走向海外也许存在更多的机会。

感谢VisionMobile（http://www.visionmobile.com/）的授权，如需转载，请与《程序员》联系。

颠覆Craigslist的机会与陷阱

文 / 骆轶航



Craigslist虽然看似陈旧，但巨头之力仍难轻易撼动

好像任何试图蚕食或颠覆Craigslist的创业项目都会获得风险投资者的强烈兴趣和至少百万美元的早期投资，至少现在的硅谷正是如此。

Craigslist看上去与我们这个时代越来越格格不入了：它存在的时间太久（成立于1995年），界面外观几乎不曾有过重大更新，网页上密密麻麻的分类广告链接的体验和报纸上的中缝广告相当接近。尽管Craigslist最近也推出了iOS和Android的移动客户端，但它的UI设计放在随便一个应用程序开发商那里都会被判定是垃圾。更重要的是，Craigslist似乎并不以此为意，它甚至连搜索分类信息时的过滤条件都不愿意设计得更复杂一点，更别提采用滑动条来过滤信息和产品这种越来越流行的设计了。

问题是它仍然很赚钱：每年5亿美元。在我看来，Craigslist是一个致力于维持最差用户体验却因此受益的公司。这也证明了即便是在加利福尼亚州这个颠覆者不断出现的地方，强大的巨头在创新者面前形成的羁绊仍然无处不在。

但硅谷的投资者会鼓励那些破坏Craigslist的创新（想想另一片土地上VC“避开腾讯”的劝诫）。Craigslist的CEO Jim Buckmaster说：“Craigslist只会提供用户最需要的服务。”这句话是不是听着很耳熟？没错，一个叫Paul Graham的人这两年也在重复这句几乎一字不差的话：“Make something people want”——这句话已成为Paul Graham创办的孵化器Y Combinator（以下简称YC）的座右铭。

因此，这不是巧合，YC投资和孵化的大量项目事实上都在挖Craigslist的墙角。Paul Graham本人也罗列过容易被YC盯上的若干个领域，其中之一就是“Craigslist的挑战者”。

不过YC到现在为止也没真正投资和孵化过一个能全面取代Craigslist的项目，倒是把一连串在局部领域试图蚕食Craigslist的创业公司都收入麾下。其中最成功的是Airbnb——这个将自己的房产短期转租给旅客的在线服务估值已接近20亿美元。其实，它挖的恰好是Craigslist在房屋租赁分类里短期租赁（sublets/temporary）的一个微不足道的角落而已。

在Airbnb大获成功之后，YC孵化了一系列类似的公司，并帮助它们拿到了数目不菲的启动融资：42Floors是一家帮助中小公司寻找合适的商业办公场所的垂直搜索和交易平台，在从Y Combinator毕业后5个月拿到了500万美元的种子期融资；另一家YC的孵化公司Your Mechanic提供的是基于网页和移动应用程序的上门修车服务，将维修工信息数据化和透明化，刚刚拿到180万美元的早期投资；此外还有二手货比价网站Priceconomics和二手车搜索网站Carsabi，也分别

拿到了大笔的融资，它们也都分化了Craigslist在汽车修理、二手车购买、商业楼宇租赁以及二手货物交易等领域的用户。

这并非YC的专利。其它团队也在这些领域成为被关注的对象，并为此拿到了投资。例如基于地图的租房搜寻工具PadMapper、Zillow和Zumper，以及提供医疗保险管理的工具Careticker等。

而最被认为可能颠覆Craigslist的是一家叫做Zaarly的创业公司——他们2011年2月在洛杉矶展示了产品原型，并在一个月之后的SXSW上发布。Zaarly是一个侧重于移动的“反向Craigslist”服务——用户通过Zaarly应用程序贴出需求，希望以什么样的价格获得，获得需求的地点以及何时候需要，Zaarly就会通过平台分享的需求信息，以得到其他用户的响应。Zarrly至少在两点上撼动了Craigslist：它把从卖方出发的需求转移到了卖方，这有助于让交易信息更加透明化；它还是一款更倾向于支持移动设备和基于地理位置的产品。尽管Zarrly成立还不到两年，可前eBay CEO惠特曼（Meg Whitman）还是加入了Zarrly的董事会，它也获得了1400万美元的投资。

其实这些试图挖Craigslist墙角的创业项目都可以用一句话来概括它们正在做的事：Disrupt the main street（颠覆主干道）。它们把那些散落在街边的与人们生活相关的各种服务拣起来，把它们移动化或社交化，用更漂亮和体贴的产品细节设计包装起来推送给人们。

但好像它们谁都没能真正成为Craigslist的破坏者。Craigslist依然维系着糟糕的界面、越来越多的骗子信息和整体上每况愈下的用户体验，但它仍然是赚钱机器，反倒是这些不断挖Craigslist墙角的挑战者中的很多都迅速寂寂无闻了。

原因是什么？几天前我见到了这届TechCrunch Disrupt的创业公司竞赛冠军，也是上一届Y Combinator的孵化公司Your Mechanic，这是家自称“Car Repair2.0”的公司。我问他们现在有多少个体维修工人加入了Your Mechanic的网络，创始人Art Agrawal对我说他们现在只有8个维修工，仍然处于很早期的阶段，而且服务范围仅限于旧金山和湾区的若干城市。我问他怎么才能让这项服务迅速拓展到更多的城市。Art说这将主要靠口碑传播，并解释说他们吸引个体维修者加入网络将制定严格的考量标准和背景调查。但这仍然不能解释他们如何实现规模化的根本问题。

在Your Mechanic的例子上，让维修工的团队规模化的过程对普通用户而言是一个获取大量数据的过程。对此前没有任何现成数据积累的创业团队来说，获得原始的能让服务迅速通过大量数据改善用户体验的过程，却往往要通过大量线下运营的“dirty work”来实现。

而有点儿尴尬的是，让这些致力于颠覆Craigslist的创业团队获取海量数据的最重要方式仍然是抓取Craigslist的数据——这甚至已经成为它们中很多团队的数据来源。在Airbnb早期，大量用户贴在Craigslist的短租广告会直接关联到自己的Airbnb页面，而其他大量类似服务也都从Craigslist抓取和索引了大量的内容。

你觉得Craigslist会对此视而不见吗？在Airbnb迅速崛起后，Craigslist立即宣布了一项政策：任何人张贴在Craigslist上的招短租信息如果包含Airbnb字样都将不予显示，并拒绝解释原因。而且Craigslist的封杀举措还迅速扩展到了其他的类似服务上——今年以来它一直在给Y Combinator旗下的相关公司和其他团队发律师函，要求它们不得再索引Craigslist上的条目，涉及的公司包括二手车搜索工具Carsabi、房租租赁平台PadMapper以及商业楼宇搜索交易平台42Floors。

对这些尚未形成规模的公司来说，被Craigslist这样一个本地生活服务的数据垄断者禁止抓取数据显然是个巨大的伤害。这就是这些创新者的悖论。

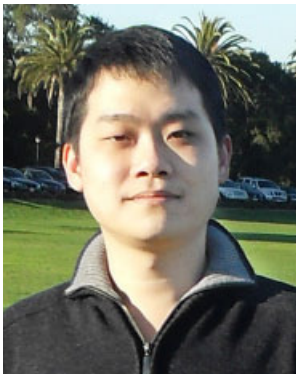
不过似乎Zaarly仍然是一家值得指望的吞噬Craigslist的公司——不过它似乎也面临着一个迅速规模化的问题。从上线到现在，累计在上面发布的需求金额尚不足1000万美元（并非实际交易额），这只相当于Craigslist一周的交易量。



骆轶航

《第一财经周刊》主笔，驻旧金山硅谷代表。关注硅谷最热门的创业趋势和成长型公司，侧重移动应用开发、云计算、社交游戏等领域的发现与报道。

游戏创业团队的技术选型



王哲
毕业于南京大学，2010年开始经营维护Cocos2D-X开源社区至今，负责《捕鱼达人》系列游戏的跨平台支持。

Cocos2D

Cocos2D的稳定性、可商用型和流行程度已无需证明。目前App Store中国区付费总榜前三十名约有50%是基于Cocos2D开发的，几个月来长期如此。因此：

- 只要你是做2D游戏就应该用Cocos2D引擎；
- 根据发布目标平台和团队擅长的编程语言选择不同Cocos2D分支，开源社区尊重每位程序员自己的喜好和口味；
- 不用担心法律和授权费用的问题，Cocos2D及其集成的第三方库都是非GPL/LGPL的；

Cocos2D家族包含了一系列不同语言、不同渲染方案的多个分支，目前稳定成熟、有商业游戏发布的主要有三个：

- Cocos2D-iPhone，用优雅的Objective-C语言进行游戏开发，最近出了JavaScript绑定，游戏只能运行于iOS设备上；
- Cocos2D-X，用经典的C++进行游戏开发，加以Lua绑定和JavaScript绑定，游戏可运行于iOS、Android、Windows Phone 7&8、Windows 8 Metro、BlackBerry 10、bada、MeeGo、Linux、Mac OS X等手机和桌面系统上；
- Cocos2D-XNA，用C#进行游戏开发，可运行于Windows Phone 7&8之上。

还有一个分支，虽然目前尚无大型商用游戏，但未来相当看好：

- Cocos2D-HTML5，基于HTML5规范集开发，采用JavaScript语言，游戏可运行在Chrome、FireFox、IE10、Opera等支持HTML5的浏览器内。

Cocos2D系列引擎对不同平台和开发语言的支持如表1所示，注意：虽然有不同分支，但只要是在同一个大版本号1.x或2.x下面，API接口是完全一样的。

表1　Cocosd系列引擎对不同平台和开发语言的支持

	Objective-C	C++	Lua	JavaScript	C#
iOS	√	√	√	√	
Android		√	√	√	
Windows Phone 7					√
Windows Phone 8		√			
BlackBerry		√	√		
Windows XP/7		√	√	√	
Windows 8		√			√
Linux		√			
Mac OSX	√	√			
Web Browsers				√	

采用与开发平台自身相同的编程语言是个不错的选择；但按照目前的趋势，跨平台已经成为一个基本考虑点之一，因此选择具有跨平台能力的开发语言，会让游戏开发和运营更轻松愉快。所以开发语言的选择上，大致可以这么考虑：

- 系统原生语言：例如在iOS上选择Objective-C，在Windows Phone上选择C#，开发时能很方便地集成SDK上的各种功能——原生UI框架以及ShareKit、ASIHttpRequest之类的第三方库，且易于调试。但游戏无法跨平台，因此现阶段不是太推荐；
- C++：保持高性能的同时可以跨不同平台，调试方便，但开发进度偏慢，集成部分第三方库需要一次语言转换，适合技术功底比较强的小型创业团队，在开发中小型游戏上有优势；
- Lua、JavaScript等跨平台脚本：可以在运行

性能和开发速度上取得一个折中，缺点是调试和集成第三方库不易，适合有一两个技术高手能驾驭语言转换层（需要二次转换，如Java>C++>Lua），然后招聘脚本程序员大量堆逻辑的中型创业团队，而且团队里程序员越多，使用脚本带来的增益越明显。此方案在开发大型游戏上有进度优势。

在工具方面，Cocos2D和多数开源社区一样是个集市，没有建造大教堂思路下集中控制的一站式解决方案，因此你需要从不同软件提供商手里购买针对不同使用目的的工具，基本覆盖了游戏开发过程的方方面面，比较流行的有：

- CocosBuilder是目前最好的UI编辑器和动作编辑器，拥有开源免费MIT许可。在2.1版本之后加入了大家翘盼已久的时间轴动作编辑功能。其作者Viktor Lidholt已被Zynga的Cocos2D团队收编，因此CocosBuilder很显然会是整个社区最重要的编辑器之一；
- 71squared的Particle Designer，必不可少的粒子编辑器，8美元一套，还不到买一份肯德基全家桶的钱；
- mapeditor.org的Tilemap Editor，这个是开源免费的；
- Texture Atlas打包工具如Texture Packer、Zwoptex；
- SpriteHelper、LevelHelper系列。

除此之外，还有一些比较新潮但用户不多的工具，例如CatHide.com，让你在一个代码编辑器里完成基于Cocos2D-X的多平台开发调试部署；还有蛋疼到碎的iTileMaps，让用户在iPad里面编辑tilemap地图，你可以买一套送给美工同学，然后享受他想杀死你的眼神。

社区支持一直是Cocos2D引以为傲的事情。只要你礼貌地在Cocos2D-iPhone或Cocos2D-X论坛里询问，总能得到来自世界某个角落里热心开发者的解答。除了官方论坛，国内还有39个Cocos2D QQ群，百度文库里4534篇Cocos2D相关文档，SlideShare上196份相关PPT，CSDN下载区701份相关资源，以及海量的技术博客，都会是你进入Cocos2D开源世界很好的学习资源。

总结Cocos2D开源引擎适合于这样的创业团队：

- 希望和Zynga、Glu、TinyCo、4399等使用Cocos2D的顶尖游戏公司站在同样的技术起跑线上；
- 希望掌握产品的每个细节，且团队中有靠谱的程序员；
- 希望能天马行空作出自己游戏独一无二的效果，不喜欢被闭源产品束缚作出同质化产品；
- 希望针对中国市场特殊性，能不费力地搞定千元智能机上的性能问题。



王楠
2005年毕业于浙江大学并担任上海育碧软件关卡设计师。2008年加入德国YAGER工作室。2010年联合组建了aBit Games独立游戏开发团队。

Unity

刚结束的Unity开发者大会（Unite2012）上，Unity制作的移动平台游戏阵容达到了有史以来的最强点：畅销游戏数量正在步步紧逼Cocos2D一系，而3D顶级大作的成色也不逊于UDK引擎的作品。这个阵容包括2D和3D的益智、平台、射击、战略、竞速等几乎所有的主流游戏类型，引擎的成熟性和多样性已不需要更多证明。

不过对于创业团队来说，Unity并不是必然的选择。开发者首先要面临的问题是可观的授权费用，Unity虽然有免费的基本版，但想要在iOS或Android平台上发布则各需要400~500美元每人的授权费用。此外还有让人头疼的Unity Pro授权，包括减少App尺寸、自定义Logo画面和编译插件等诱人特性——但想要在iOS或Android上使用这些特性就要额外付出每人3000美元每人。

第二个问题也同时是Unity最大的优点：Asset Store市场上种类众多的中间件和资源。如果你愿意为快速整合游戏内置购买功能而付出60美

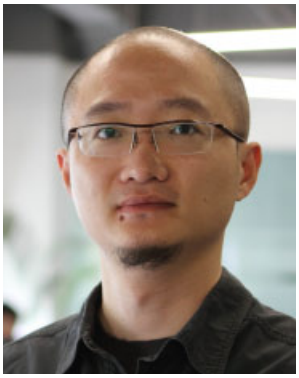
元的话，那么很多烦琐的功能整合都可以在很短的时间内实现。此外还有可视化编程工具，各种美轮美奂的材质、人物、动画。Unity的Asset Store对于有一定资金支持的团队来说是缩短开发流程，甚至节约成本（相对于外包需求）的重要方式。但反过来看，如果你的团队没有条件或者不喜欢使用第三方工具和资源，那么很多功能和实现在Unity里就需要很多额外的工夫了。举例来说，和设备Native Code通信方面，Unity就会被Cocos2D完爆，Cocos2D开发者只要调用苹果官方库就能实现的功能，在Unity里却要兜几个圈子。

最后还有Debug方面的不足，Unity游戏在iOS上运行需要使用Mono的架构把游戏代码编译成汇编，然后才能在Xcode里使用。因此Unity移动平台游戏无法直接进行真机Debug，只能配合Log和经验来摸索。所以在遇到触屏操作问题和移动设备专用的服务问题时就会特别棘手。

尽管问题多多，很多团队还是把Unity作为了移动开发的首选，这和Unity公司重视技术和开发者群体的运营方式有很大关系。整个Unity社区的活跃程度和大量商业化带来的技术支持专业化在目前的市场上看都是独一无二的。加入Unity开发者的阵营不仅能得到一个强力游戏引擎，随之而来的还有大量技术知识来源，配套的新技术服务，求职创业两不误的人脉群体。

想要最大程度发挥Unity的优势，我建议开发者遵循一个“最速出原型>使用现成工具补充功能性的不足>重构程序结构>完善完善再完善”的流程。Unity的原型制作流程是非常高效的，而且有着对于美术和设计人员非常友善的开发环境（有时需要借助一些自定义的编辑器工具），但3D引擎架构的复杂程度决定了如果不及时更新工具或重构代码的话，接近成品时就会有日益臃肿直至一团糟的可能。只有每隔一段时间整理一次游戏架构，才能保证游戏接近完成时能够随心所欲的完善，以及发售后更新需要的可扩展性。

总结：Unity移动游戏开发适合有一定资金支持的小型到中型团队，如果你重视工具链条的构建（或者购买）以及游戏架构的整理和积累，开发速度和游戏质量都会提升得很快。



刁劲翀

2006年加入英佩游戏并担任副总裁，主要负责虚幻引擎在中国和东南亚的授权、市场及商务拓展工作。同时，他还是GA游戏教育的执行董事和创办人之一。

Unreal

提到虚幻引擎（Untral），在国内开发者中可能有这样一种认识，它太重、太复杂，不是非常适合初创团队的需求。但实际上，虚幻引擎有不少特性是能够很好地帮助创业团队的，使用虚幻引擎获得成功的创业团队也不少。

例如开发《无尽之剑》的ChAIR团队，他们从创业伊始就一直在使用虚幻引擎。开发《无尽之剑》之前，ChAIR曾做过两款成功的横版射击类游戏《Undertow》和《Shadow Complex》，每款都有数百万美元的收入。成功的作品也让ChAIR受到了Epic Games的青睐，最终被Epic所收购。即使在开发《无尽之剑》系列期间，ChAIR依然保持着非常小，只有13个人的团队规模。《无尽之剑》系列的游戏在iOS平台上的收入早已超过3000万美元，推出时被认为重新定义了移动游戏在画面品质上的最高标准，而其第一代产品正是由这13人的团队在非常紧张的5个月时间里完成的。虚幻引擎之上的高品质表现和极高的开发效率也是至关重要的成功秘诀。

对一个希望以品质取胜的游戏来说，最终的画面效果能不能充分表现必要的意境和情感，最终的玩法能不能带给玩家最佳的反馈和体验，都可能影响游戏的最终成败。但游戏设计时的预期和最终呈现的效果在开发过程中往往会有不小的差异，需要美术、程序和策划人员反复的沟通和修正。但无论是当面的交谈还是文档，都没有能力准确表达游戏设计者（美术

或者策划）的意图。这种不准确的沟通形式会导致开发过程中的大量返工，增加迭代次数，从而引发项目延期、成本上升等让人头疼的问题。我们认为为每个参与游戏设计的人提供可以自己动手实现自己的需求，是解决这个问题的一个有效手段，也是虚幻引擎的一个主要的设计思想。虚幻引擎的开发团队投入了非常大的精力来开发便捷易用的工具。也正是在虚幻引擎完善的，真正的、所见即所得的可视化工具的协助下，《无尽之剑》开发团队仅用了3周时间就完成了核心玩法的设计，用了4周时间就确立了美术风格和开发技术标准。而且后期的开发中，也没有因为核心玩法和美术标准方面的因素导致返工或拖延。

虚幻引擎另一个可以帮助开发者降低开发成本的重要特性就是支持跨平台开发。目前虚幻引擎已经支持了包括Windows、Android、iOS、Flash、Mac OS等11个主流的游戏运行平台。开发者可以完全用一个项目的开发周期和成本，开发瞄准多个发布平台的游戏。美国有一个叫Trendy Entertainment的初创游戏公司用虚幻引擎开发了一款叫《Dungeon Defenders》的多人在线动作类游戏，这款游戏先后发行了PC、Xbox360、PS3、iOS、Android、Flash的版本，取得了相当不错的成绩。最近，在使用虚幻引擎开发移动平台游戏的开发者中，有了一个新的趋势，就是同时发布iOS和Android的版本。例如Gameloft的《Wild Blood》和Phospher的《HORN》。虚幻引擎在跨平台方面的便利性，也让这个有利于开发者的趋势变得越来越显著。

最后，再谈一下创业团队都很关心的学习成本和技术支持问题。坦率的讲，在易于上手方面，虚幻引擎并不是做得最好的。但大家一定都会有亲身体会，一个相机自带的修图软件通常是很容易上手的，但如果要达到商业级的图像处理效果，使用专业软件是不可避免的，而功能更全面也更强大的专业软件的学习成本一定也会更高。不过这并不意味着我们不重视开发者的学习成本，在虚幻中有很多细节的考虑都是从易于开发者上手的角度出发的。例如我们的关卡编辑器有两套鼠标与快捷键操作方式，一套与3ds MAX一致，一套与MAYA一致。因此无论关卡设计者熟悉的是MAX还是MAYA，都有适合他最快上手的选择。另外，我们在国内数十人的工程师队伍，也给用户带来了培训和技术支持上的最大保障。

对于创业团队选择第三方技术，我觉得根本的原则还是要选适合自己的，适合自己的能力，也适合自己的项目。如果一个3D游戏项目是希望通过高品质取胜，又希望拥有非常高效的开发流程，那虚幻引擎可能会是一个合适的选择。



德义
从2007年毕业开始，先后制作了20余款Flash单机游戏，2011年创建独立游戏团队“3000GAME”，其首个独立游戏《被错过的天堂》一度夺得台湾App Store付费榜榜首。

Flash AIR

相信大家应该对Flash比较了解，但对于Flash AIR可能就不是那么熟悉了，尤其前阵子Adobe宣布将移动版Flash Player停止维护及开发，就让大家误认为Adobe Flash将退出移动平台。

但实际情况又是如何的呢?

Flash并不只有Flash Player、Flash IDE等一些大家耳熟能详的产品，实际上“Flash平台”是一个统一的称呼，它其实包括五个方面：运行时（Flash Player和AIR Runtime）、语言（ActionScript 3和MXML）、框架（Flex、TLF、OSMF等）、开发工具（Flash Pro、Flash Builder等）、周边方案（Alchemy等）。而Flash AIR正是Adobe对于移动平台的定制的重头策略。

Flash AIR最开始是Flash对于桌面应用的支

持，在移动开发火热之后，AIR开始转变策略，转而为Flash内容的移动App打包进行支持，因为Android设备上允许安装AIR运行时，所以采用AIR打包为Android设备部署。Flash应用就变得顺理成章（虽然和桌面相比，内在运作机制还是很不同的）。当iOS拒绝了Flash Player进入之后，AIR又担负起了将Flash应用部署到iOS设备的重任（因为iOS不允许安装运行时，AIR针对iOS的打包策略和Android设备上又是完全不同的，需要直接编译出本地可执行代码，但为了统一平台起见，仍归属在AIR的范畴之内）。

现在，Flash AIR的核心策略显然已转变为移动开发，而今年Adobe对AIR的开发支持更是空前的强力。加上Adobe在HTML层面对PhoneGap的投资，可以看出Adobe在移动策略上采用了Adobe AIR（for Flash）和PhoneGap（for HTML、JavaScript Developer）的双保险策略。

下面我就主要从开发时程、效能、跨平台性来说来说一下Flash AIR对于游戏开发的优缺点。

■ 开发时程。Flash多年经验积累下来，为广大开发者准备了一套非常完备的开发流程，从图形动画制作工具Flash Pro到编码工具Flash Builder的每一点，无不体现出“可视化”、“便捷”等特点，让使用者可以最少编写代码来达到想要得到的效果，更加上不可替代的动画支持，让整个开发过程锦上添花。而AIR同样享有这一切。随便写一个小特效的话，Flash可能只需要几行代码，而其他语言则可能需要几倍的代码量。

■ 效能，对于Flash而言，效能一直是其非常褒贬不一的地方。事实上，Flash的效能完全取决于使用者的优化水平，因为Flash为使用者提供了尽可能多的图形图片处理方式，这就需要使用者有能力去衡量每一种方式的适用情况。更加上Flash Stage3D已经可以在移动平台调用GPU，Flash的效能就更加得以提高。在iOS平台上，如果优化得当的话，是可以接近Open Canvas的效能的。

■ 跨平台性来讲，Flash也是十分优秀的，PC、Android、IOS、BlackBerry、Windows Phone(未来支持)，全都可以用同一个SWF文件去发布，这意味着我做了一个应用就可以轻松发布到其他平台，也省去了移植的一些工作量。

对于以上的几点综合来看，我决定选择使用Flash AIR技术。而对于Open Canvas与Java而言，虽然性能出众，却没有很好的跨平台性支持，而HTML5则由于还是新技术，可能存在许多隐患，制作流程上也显得不够成熟。

事实证明，我的选择是对的，一个版本可以成功打包成很多平台的应用这让我欣喜若狂，而且对于游戏来讲，一般的2D游戏AIR运行起来也是没有压力的，效能也不用很担心。

下面说说关于优化方面需要注意的事情，前文说过，Flash AIR技术需要比较好的图形优化能力，其实就是对于图形图像算法的理解能力，精确运用滤、颜色叠加、动画、动画层次以及内存等元素可能是需要开发者注意的。此外还有关于内存的处理，Flash技术把内存做了很智能的处理，不需要用户太关心内存回收，但是也因为如此，内存回收变成了一个不可控的元素，我们需要积极地将用不到的对象的引用消除以保证能够尽快回收。

自从ActionScript出现以后，Flash便成为了小游戏的最佳选择，快速的制作流程以及丰富的动画支持变成了Flash技术的利刃。这么多年下来，众多Flasher创造了不计其数的Flash小游戏，后来还有很多优秀的游戏转移到了其他大型平台，并且销量非常好。这些年，Flash积累了无数的经验及产品，而这些所有的东西都会直接移植到AIR技术上，就让AIR制作游戏更加如虎添翼。

选择何种技术很重要吗?

很多游戏制作者都喜欢关注游戏使用了哪些

技术、用了什么框架等，但事实上玩家完全不会在乎哪一款游戏是用了什么引擎、框架、设计模式，甚至不是很在乎到底是谁制作了这款游戏。

作为游戏开发者，我们要把时间更多地投入在如何让游戏更受大家喜欢，如何让游戏更有内涵或者更耐玩，而对于收费游戏来讲，我们可能要考虑如何让大家很开心的付费。要知道，技术不怎么样但很卖座的游戏其实是不在少数的。

对于技术而言，不用非要追求最好，而是恰当就好，选择最适合的最利于目前开发的项目的技术是最佳选择。

千万不要小看市场

经常有人说：“《Angry Birds》很简单，我们也能做！”之后就几个人开始做一款山寨的小鸟、《Fruit Ninja》，又或是一些别的什么游戏。我个人对于这种尝试还是比较赞成的，因为一旦做了就会获得经验，这样对开发者个人而言是只有好处没有坏处的。

但我想说的是，千万不要小看市场，我们的市场很挑剔，每一个作品都追求精益求精，并不是随便哪个游戏都能够赚钱，不是随便凭借山寨一个什么产品就能够成为“高富帅”。这个圈子里的赚到钱的人都是在非常努力地做产品，当然不排除有些开发商采用一些投机的方法去谋利，但如果想要真正在市场站住脚的话，还是需要一定的积累跟实力的。

事实上，有很多开发者连每年支付给苹果的开发者授权费都赚不回来，因此做游戏别太冲动，一切三思而后行。

精准把握玩家行为

我觉得我们需要花费精力去分析玩家行为，这样我们就能对玩家对于某些操作或游戏模式的反应做出正确的判断，也能帮助我们设计出更人性化的游戏，增加游戏的用户黏性，从而让更多的玩家不仅能够接受我们的游戏，还能够长期玩我们的游戏，这样自然也就带给我们想要的利润。

踏实做游戏

很多团队做游戏时就会想到以后会赚多少钱、如何发达等，但只要一旦这样想，就容易走入某些奇怪的误区里，我们还是需要冷静去面对游戏制作的每一个部分，这样才有更多的机会脱颖而出。

团队向心力

团队向心力是比较不好处理且经常出问题的环节，它可能包含了比较多的可能性，例如分配不均，或者内部分裂，又或是由于成员能力不足等原因带来的问题。可能任何一点小小的问题都会导致整个团队分崩离析。

3D加速技术

Flash AIR技术虽然到处可见，但其实大家对这个技术并不是很了解，网上的评论更是曲解的离谱，实际上Flash的发展非常迅速，加上Adobe对3D的投入就更加让所有开发者欣喜，遗憾的是，在我们的独立游戏《被错过的天堂》中，并没有使用3D加速技术，主要是因为游戏制作时3D加速技术还没有被应用在移动平台，但在将来的游戏开发中，相信我一定会尝试3D加速技术的。

靠谱的渠道在哪儿?

文 / 张礼镜

我从海军特种兵转业之后，从2002年开始从事中国手机软件营销工作。回想2002-2005年间中国的手机软件行业，基本上只有一些简单的功能性软件，采取单向手机厂商收取一定的授权费用的商业模式盈利；2005－2009年中国的手机软件行业是SP乱战时代，这个时期违规扣费现象严重，大部分采取免费或是付费给手机厂商，后续向用户收取一定的SP增值服务费，而且许多无良服务商会不定期暗扣用户话费；2009年后SP行业受到政策与行业内的监管与谴责，特别是伴随着Android/iOS生态系统的高速发展，更多的开发团队开始投入到智能机的开发。在短短的3年时间内，开发者数量已经达到了百余万，开发出的软件和游戏作品也达到上百万件，而在这种表面火爆的移动互联网创业潮中，真正实现盈利的只有10%左右，大部分的开发团队在推广以及运营过程中遇到瓶颈。什么样的推广渠道才是最符合开发者自身的？怎样的推广渠道才是最有效、低成本的渠道？这一直是业界探讨的话题。本文中，我将对从业10年来的营销推广经验进行总结，对主要针对Android产品的推广渠道进行优分析。

线上推广渠道

广告平台

优势：起量快，效果显而易见。

劣势：卸载率较高、沉默用户多。

推广成本：成本较高，以目前主流平台为例。CPC在0.35~0.8元；CPA在1.5~3元（联网激活或一键注册）。

业内公司：Admob、多盟、微云、有米、亿动、架势等。

发展方向：广告平台目前都生存得很困难，这个在移动互联网上最能看清楚商业模式的产品将面临一次大的变革，逐渐由原来的绩效类广告向品牌广告发展。广告的展示形式也由原来的图文向富媒体发展。

操作难点：因为推广成本高，不利于创业融资前的团队采用。

综合评级：★★★★

网盟

优势：量巨大，短期见效快。

劣势：卸载率高、沉默用户多、智能手机覆盖少。

推广成本：成本适中，但虚假繁荣。

发展方向：功能机上以WAP方式传播，经常有擦边诱导用户点击的广告，逐步走向没落。

操作难点：作假严重、业内公司众多、质量参差不齐。

综合评级：★

软件商店

运营商商店

优势：用户基数较大，可以将商店预装到定制机，运营商有许多第三方没有的能力，例如用户号码、用户IP、用户消费能力以及习惯，有着庞大的用户数据发掘系统，可以针对性地去推广和推荐产品。从2012年春节后开始，运营商逐步放开许多资源，其中最好的资源就是计费渠道资源。

劣势：国企作风，决策比较慢，用户下载量不如第三方商店。



推广成本：基本上是零成本，如果是精品产品，运营商会有补助以及扶植。

业内公司：中国移动、中国电信、中国联通。

发展方向：运营商2010年之前是完全强势，控制着终端、计费、流量等整个产业链。在智能机时代，因为进入者多，很可能造成运营商沦落为下载的一个通道。未来运营商也会利用自身的优势加大对开发者的把握。

操作难点：需要有专门的运营商团队负责与运营商沟通，沟通时间成本高，如果针对某个项目需要出完善的方案。

综合评级：★★★

第三方商店

优势：进入早，有部分用户积累，大部分由传统互联网企业建设，完全以互联网的理念经营，合作态度开放，反馈问题迅速，用户活跃度高。

劣势：全国有一百多家第三方商店，竞争激烈，淘汰率高。除卖广告位外，没有比较好的商业模式。

推广成本：如不需要广告位以及推荐，推广成本基本为零。但每天有大量的产品上架，又会造成产品很快被淹没下去。

业内公司：安智、安卓网、机锋、优亿、应用汇、N多、木蚂蚁等。

发展方向：商店多、同质化严重，都在抢占入口，主流商店都在开放和封闭中徘徊。接下来竞争残酷，优胜劣汰，剩下几家有特色的。

操作难点：商店众多，需要专门团队维护，需要有大量的素材、测试等工作进行。

综合评级：★★★★

手机厂商商店

优势：在自己品牌的手机里预装商店，用户把握能力较强。华为、中兴等大厂商会有对中小开发团队的资金扶植。

劣势：正因为只能在本品牌手机预装，这反而是劣势。手机厂商的商店运营经验没有第三方商店丰富，并且顾及品牌，需要正版或授权内容。

推广成本：如果不需要广告位以及推荐，推广成本基本为零。

案例分析：2011年《最后的防线》与某知名手机厂商共同举办“寻找明星开发者”活动，可以对开发团队进行包装，并且有一定资金的扶植。

业内公司：智汇云、联想乐商店、HTC市场、MOTO智件园、魅族市场、OPPO NearMe等。

发展方向：缺少好的营收模式。做，投入人力物力但没有很好的收入——有点痛苦；不做，总感觉比别人少点什么——不能掉队。

操作难点：商店众多，大厂商有自己的商店，中小品牌厂商部分采取将商店外包给斯凯、悠悠村等SP运营。需要较多的运营人员。

综合评级：★★★

社区推广

优势：目前主流的智能手机社交平台，潜在用户明确，能很快地推广产品。利用好友机制进行推荐，可以大大地提高转化率，有统计显示48%的用户是通过好友间传播完成的。特别是在病毒式传播逐步增多的情况下，好友间传播有一定的信用度。

劣势：有部分社区平台需要采取独家方式，制约了产品开放性合作。

推广成本：基本采取分成方式，合作流程快。

案例介绍：2010年7月通过微云社区上线的《非诚勿扰》交友软件，在微云社区中用户自传播自推广，一个月用户量发展到了32万。

业内公司：微云、九城、新浪、腾讯等。

发展方向：增加互动，完善用户、充值、虚拟货币系统，集成自有或合作的广告平台，为开发者提供一体化的服务。

操作难点：要突破独家平台合作的模式，需要有较强的商务沟通能力。

综合评级：★★★★★

积分墙推广

优势：起量快，效果显而易见。

劣势：大部分用户都不是真正因为软件好、有需求而下载使用的。由于用户留存率不高，造成推广成本加大。

推广成本：比广告平台单一成本低，大部分采取CPA的方式，价格从1~3元不等。但以活跃用户等综合成本考量，成本偏高。

案例介绍：2011年12月一款游戏加入了推广墙平台，预装到了某品牌手机中。手机上市四个月，销量达到40万台，通过推广墙推的软件有28万用户激活。

业内公司：Tapjoy、微云、万普、有米等。

发展方向：用户的细分化，可以根据用户的喜好去推荐产品，这样才可以更好地提高用户活跃度。

操作难点：适合大型、有资金的公司。需要尽快发展用户的团队。

综合评级：★★

海外推广

优势：海外用户付费意识强、用户转化率高、单一用户ARPU值高。

劣势：异国运维成本高，当地没有推荐、推广等资源。

推广成本：媒体宣传以及运营成本偏高。

案例介绍：2011年6月，微云将2款游戏进军韩国市场，到年底收入总排名在39位，同类游戏排名第一。其中有许多运营的技巧。

发展方向：未来中国更多的精品游戏需要进入海外，能为开发者赚取更多的收入。目前以韩国为例，运营商有五种计费方式，基本上可以满足所有用户的支付。

操作难点：适合与异国当地文化符合的游戏以及应用产品的团队进入。需要前期详细的评估。

综合评级：★★★★

线下推广渠道

手机厂商预装

优势：出厂就存在，用户转化率高，最直接的一种发展用户方式，对公司以及产品品牌有一定的帮助，是每个开发团队都想抓住的最有效的渠道。

劣势：1. 用户起量周期长：从提交测试包测试－通过测试－试产－量产－销售到用户手中需要3~5个月时间；2. 厂商排产表机型众多，需要针对不同的硬件配置进行适配，机型多造成每款机器定位不同，需要针对不同定位推荐不同的产品；3. 厂商人员结构层级多，需要有业内有经验的人员寻找关键人，以达到高效合作的目标；4. 如果是同质化严重的软件，厂商有可能需要向开发商收费预装。

推广成本：应用：根据应用类软件同质化程度、大小、受众群体不同，成本不同，基本上厂商首选是

预装量付费价格从0.5元~1元不等，如果CPA的方式一般价格在1.5~4元不等；游戏：单机产品一般采取免费预装，后续分成的方式，网游产品目前出现了一些CPA的方式价格在2~3元之间。

案例介绍：1. 国际品牌：2011年《最后的防线》等三款游戏授权给某国际厂商，其中一款游戏授权费用高达5万美元，同一款手机其他5款游戏是由某运营商推荐，每款游戏授权费用为800美元；2. 国内品牌：《疯狂打地鼠》、《斗地主》、《最后防线》、《巅峰狂飙》等一系列产品授权给国内知名品牌厂商，后续采取分成方式合作。

业内公司：华为、中兴、酷派、波导、TCL、海信天迈、小米、魅族、百度、阿里、盛大、360、、G-five、酷派、联想、金立、OPPO、BBK、夏新、首易等。

发展方向：厂商已经明显感觉到单纯的靠硬件利润很有限，特别是苹果模式的成功让更多的国内外品牌厂商看到服务的重要性，纷纷建立软件商店。

操作难点：品牌众多、人员层级多、产品项目多，需要有专业的团队进行针对性的推荐与维护关系。

综合评级：★★★★★

水货刷机

优势：起量快，基本上2~4天就可以看到刷机用户，数量大，基本上一天可以刷几万台。

劣势：重刷现象严重，基本上一部手机从总批到渠道到店面会被刷2~4次，造成推广成本巨增，用户质量差，不好监控。

推广成本：基本上刷机单一软件CPA在1~2元，包机一部机器价格在5~10元之间。

案例介绍：2010年下半年到2011年中是刷机最为火爆的时段，许多渠道以及开发商都加入其中，但从去年下半年开始刷机渠道的弊端显露出来，单一用户成本甚至可以高达60元。第一批刷机商有一部分转型成为提供刷机设备的供应商，一部分开始向柜台控制渠道。

业内公司：XDA、酷乐、乐酷、刷机精灵等。

发展方向：刷机只是特定时期出现的特定产物，市场回归理性后刷机市场有可能渐步消失。

综合评级：★

行货店面后装

优势：用户质量高，用户黏度高，用户付费转化率高，见用户速度快。

劣势：店面众多，店员培训复杂，需要完善的考核以及奖励机制。

推广成本：基本上CPA的方式价格在1.5~3元之间，预装价格在0.5~1元之间（店面提供查询后台，可以随时查询预装手机的IMEI号以及信息）

案例介绍：2010年底与某国内知名连锁手机卖场进行合作试点，将合作的光盘放到店面进行打包销售给用户，8万套光盘激活用户达到了近一半。

业内公司：乐语、中复、长江天音、中邮、天音、苏宁、国美等。

发展方向：零售商和手机厂商感觉相同，因为硬件的利润十分有限，必须要用软件以及游戏来做增值的服务。

操作难点：需要制定出完善的培训以及考核系统，这样才能让促销员积极地来推广产品，到零售店买手机的用户基本上是以前SP时代最为优质的用户，这部分人不是玩机族，对电子产品不敏感，促销员的教育能使用户成为忠实的用户。

综合评级：★★★★★



张礼镜

微云科技CBO，负责公司战略合作及商务谈判工作。曾任香港富年电子科技有限公司终端合作总监，上海浪潮灵点软件有限公司副总经理。

如何走向海外？



杨永智
海豚浏览器创始人兼CEO。毕业于华中科技大学，曾任职于微软亚洲工程院，专注于移动端产品孵化。

海外扩张善用品牌+技术攻势

海豚刚刚获得高通公司战略投资的消息大家可能已经知道，随后我会去参加高通公司在德国柏林举办的技术会议。高通风险投资中国区总经理沈劲透露一个有趣的轶事，他说高通总裁的公子玩过海豚浏览器，早前推荐他们投资海豚。不管趣闻是否当真，海豚今年在获得国外资本的亲睐之外，在开拓海外市场上，取得了突破性进展。

今年5月，海豚开始进军日本市场，与日本第二大运营商KDDI建立了品牌战略合作关系，将大规模覆盖KDDI旗下超过3000万用户。这被外界看来是海豚目前最重要的海外扩张布局。

熟悉海豚的人知道，在创业最初首选了迈入海外之路。我们最初认为中国的智能手机市场很小而美国很大，因此想先从美国做起才可能扩展到其他国家。如果从中国做起，只能扩张印度、东南亚、非洲国家。如果把品牌在美国做起，就有机会去日本、欧洲、北美主流市场。我们决定要选一个相对成熟、有强势文化的市场——美国市场。后来海豚进入日韩市场，当别人看你的标签：硅谷品牌、美国公司，更愿意跟你合作。如果以中国公司的身份要进入日本，不借助当地资本或合作伙伴就很难进入。

2010年初，海豚选择先从美国的Android市场上起步。在发布仅17个月的时间，就在Google Android Market浏览器类下载量排名第一，在PCMAG评的最佳免费安卓应用中，海豚浏览器居首。后续用户量也都在稳步增长。

如果你决定做一款酷、与众不同、国际化的产品，同时又不具备在中国那么多的“人脉”和“社会资源”限制的话，那么海外市场是最好的选择。

靠技术实力赢得品牌认可

与其颠覆传统行业大佬，不如把优势结合，移动互联网企业没有体量和财力做硬件，选择与一家已经具备一定量产能力的传统互联网制造商合作为“取巧”之计，这不仅成为多数互联网巨头切入智能终端的首选，也将给海豚的发展刻下更加全球化的烙印。

海豚在今年的发展速度非常快。从技术角度讲，海豚曾以450分的成绩刷新了全球HTML5跑分纪录，被国际公认的HTML5 test网站选为展示案例。近日推出的海豚浏览器实验室版也出色完成了Facebook研发的Ringmark测试，成为目前世界上唯一完成第二环R1测试冲击最后一环R2测试的浏览器。海豚员工也相应

获得了业界的尊重，他们获邀起程前往高通和Facebook公司在硅谷和柏林举办的技术大会，展示出来自中国的声音。

海内外员工共同成长加深理解

虽然是一家创业公司，但我希望员工能跟海豚品牌共同成长。我们在美国、日本设有办公室，每个季度定期会派优秀工程师互访加强沟通；还在暑假发起了美国学生来华参观活动。接下来还将把机会留给一些中国学生赴海外交流。这对两国的科技文化交流非常有意义，愿意做积极的努力去推动这项公益事业。

在美国招聘资深人才需花费大量时间和精力与对方交流。这个沟通的过程也是建立信任的过程。很多优秀的人更注重平台，并非为了钱，而是为了梦想和理想。作为一个领导者或管理者，要能够让团队成员觉得你让他看得更远、学习到的东西更多，这样才能吸引更多资深、优秀人才加入，才能创造出更出色的产品。

进军韩国市场三部曲



张礼镜
微云科技CBO，负责公司战略合作及商务谈判工作。曾任香港富年电子科技有限公司终端合作总监，上海浪潮灵点软件有限公司副总经理。

要想进军韩国首先就要对韩国的移动互联网市场有一个整体的了解与把握。据韩国第一大运营商SK电讯数据统计，截至2012年7月，韩国共有手机用户5400万，其中近60%用的是智能手机。而这些智能手机基本上都是三星、LG等韩国本土品牌，这一点也为进入韩国市场做硬件适配提供了比较简单的前提。韩国的智能手机用户100%是办运营商套餐赠送的，只要每个月消费5.5万韩元就可以获得一款三星9100、I9000等比较新潮的智能手机，这点和中国运营商实行的办套餐送手机比较相似，不同的是韩国的套餐包括所有的语言、数据以及运营商一些基本的第三方业务，因此韩国用户比较习惯于随时随地用手机看电视、下载游戏等。在韩国，每个月花费在套餐以外的增值服务费用大概在3万多韩元，其中有70%是消费在游戏上面的。因此在韩国有着广泛的、良好的用户基础，比较符合进入韩国市场获取巨额利润。

那么如果想进入到韩国市场应该怎么做呢？我从以往经验来看主要分为三步进行。

进入韩国三部曲之一：本地化

进入到韩国首先要做的就是本地化工作，目前韩国的运营商给中国的企业许多优惠的举措，例如，根据产品的不同付10万~50万元人民币不等的预付费用，同时只要提交中文字库就可以免费翻译成韩、日语言，并且可以在推广渠道里进行重点推荐等。而在韩国做本地化是一件比较复杂而又烦锁的工作，这里也有许多小技巧需要掌握。例如，中国企业进入韩国要做的第一件事情就是注册一个韩国网站，或注册一个韩国当红博客的域名，同时在产品起名上要以形容词+名称的方式，这些都是做去中国化工作。因为韩国比较有民族气节，如果发现一款游戏是其他国家开发的会有抵触情绪。另外在游戏的道具关卡设计上也要与中国有所不同，韩国人玩游戏比较直接，而且很喜欢付费，所以我们在一些进攻类道具上要多设计，可以定价高一些。通过韩国运营商的帮助，以及我们自己对韩国本土小技巧的了解，基本上

完成了进入韩国的第一步。

进入韩国三部曲之二：预热

在完成了本地化工作的同时就需要对这款产品进行预热了，在韩国预热的渠道和中国有所不同，下面我们一一介绍。

■ **媒体**：韩国的游戏媒体写软文是比较方便的，而且韩国的软文写的比较负责，韩国的网站编辑会把游戏玩一遍，客观地写游戏的一些优劣，而且韩国的媒体一般不需要费用，如果产品足够好那就会有媒体来曝光。另外韩国主流媒体是Facebook和红人博客，现在韩国有几百个明星、业内的红人博主也承接游戏的测试推荐工作，这部分是需要一定费用的。

■ **广告**：在韩国有许多比较独特的广告形式，而且效果很好，例如，出租车广告，在韩国许多出租车上有类似中国一样的显示屏，在上面滚播游戏的广告片，左下角会有一个二维码区，如果乘客喜欢这个游戏直接扫描二维码就可以下载。这种形式已经逐渐成为韩国推广的一种最受欢迎的形式。

■ **店面**：韩国全国有2万多家手机卖场，因为韩国的手机都是由运营商赠送的，基本上与韩国的运营商营业厅是在一起的，如果与韩国运营商谈了深入的合作就可以把一些需要预热的游戏视频片以及测试版的游戏放在卖场的电视里和业务展示区的手机里，让用户试玩和观看，同时在节假日也可以与运营商配合，举办一起活动。

完成了以上几点就可以开始游戏的大卖和推广了。

进入韩国三部曲之三：推广

完成了本地化、预热后，就可以为游戏做推广了，游戏的推广主要分为以下几点来完成。

■ **版本发布**：这点单独出来说明，因为在中国，基本上都是以免费版本为主的。在韩国完成不是这样，首先发布的最好是收费版本，因为韩国的免费游戏每天有上万个，如果想在免费榜里排在前20位的话一天的下载量要达到10W的量级，而收费榜可能只需要下载几千就可以排在前20位了。同时，我们在发布收费版后几天可以做限免等促销活动。在韩国，版本的发布基本上采取的是“收费版本－限免免费基础版－蓝牙对战版－联网”版。要记住，在韩国发布游戏的版本一定要丰富，韩国用户喜欢付费，而他们又很挑剔产品，丰富的版本可以让用户有更多的选择。

■ **用户间奖励**：在韩国有这样的一个运营机制，就是如果集成了韩国运营商的游戏中心可以不定期在上面做一些活动，鼓励用户之间相互推荐，如果推荐成功就可以赠送一定数量的道具或是直接就送运营商的一些代金券，这些代金券可以直接到运营商那里进行任意消费。

■ **更新**：如果已经做了以上两步的推广，那么就要时刻的关注用户的评价留言，发现有任何用户的反馈要及时做出回复和版本的调整。在中国，可能一款游戏发布后随便用户怎么评价都可以不去理会，而韩国大部分都是付费购买的游戏，他们会真正地说出游戏存在的问题。如果开发者与游戏玩家有互动地去修改问题的话，那么玩家的黏度会越来越强；如果不去理会，那么用户玩一段时间就会对这家开发商失去信心而不再玩这款游戏。因此游戏的及时更新是推广中不可缺少的一个环节。

■ **店面**：同第二部第三点

通过以上三部曲，就可以把韩国游戏发行做得很完善，当然这里操作起来还会遇到许多问题需要去修正和协调。引用佛家在修行或是法会举办的三部曲一样，在开始时做好前行，我们做好本地化的准备工作；然后是正行，就是在预装和推广阶段时我们一定要把问题想全面和周到；最后是结行，那就是总结阶段，一款产品进入韩国一定要在推广一段时间后做总结，针对之前推广中的不足来不断调整。



刘勇
热酷网创始人及CEO。拥有多年的SNS及无线网络创业和管理经验。

扬长避短的日本游戏市场策略

要走向日本游戏市场，首先要对日本游戏市场有清晰的了解和认识。

从市场规模和发展速度方面来看，根据日本网络游戏协会前不久公布的市场调查报告显示，2012年日本社交游戏市场比上年增长了250%，市场规模高达2794亿日元，约270亿元人民币，再加上智能手机的迅速普及，社交游戏延续了良好的发展势头。

但日本的传统游戏市场却一直在萎缩，以知名上市游戏公司Konami的财报为例，季度总销售额下降很快，从261亿日元（3.337亿美元）降至228亿日元（2.915亿美元），季度利润也一样下滑，从61亿日元（7800万美元）降至52亿日元（6650万美元）。如果不是转向了社交游戏，Konami的损失将会更大，社交游戏在第二季度给它带来了79亿日元（1.01亿美元）的收入。Konami的传统游戏零售季度销售额从77亿日元（9840万美元）急剧跌至如今的51亿日元（6520万美元）。

日本的社交游戏以移动社交游戏为主，日本是全球最特殊的移动互联网市场，日本发达的游戏产业孕育出许多世界顶级的游戏厂商如索尼、任天堂、Capcom等，日本游戏用户生活在主机游戏、掌机游戏等环境中，养成了良好的付费意愿，但对于hard core游戏的品质也非常挑剔。在这样的背景下，移动游戏在日本形成了特殊的定位，以休闲型的移动社交游戏为主。

在休闲型的移动社交游戏中，目前最主流的游戏是卡牌对战游戏。卡牌对战游戏的流行植根于日本特殊的文化，日本教育体制从小就非常注重竞争与合作的元素，人与人之间的社交化趋势在日本社会中本身就根深蒂固地存在，这也是日本形成移动社交游戏的深层驱动因素。日本玩家从小就开始就玩实物卡牌游戏，用零花钱购买收集各种卡牌，在学校与同学交换、比较、竞争。发展到手机卡牌对战游戏时代之后，日本玩家这种喜欢收集游戏卡牌的心理被游戏公司发挥得淋漓尽致。例如当用户攒齐6张小卡牌就能获得一种特殊道具时，用户会想尽一切方式去收集自己手上没有的卡牌，这时游戏公司往往会推出一些付费抽奖活动，有较高概率抽中卡牌，吸引大批用户花钱。

要走向日本游戏市场，除了要对日本特殊的文化有深入的了解，推出的产品必须在美术上达到很高的品质，在日文本地化方面做到非常精细，对于用户隐私的保护非常到位之外，还必须理解日本移动社交游戏特殊的推广方式。以日本最成功的移动社交游戏公司GREE为例，GREE经常投入巨资去砸昂贵的电视广告，这是GREE在社交游戏上成功的重要原因。电视广告对卡牌类游戏的推广非常有效，日本玩家很多都是一边看电视一边玩手机，由于卡牌游戏通常靠游戏内的活动赚钱，精心策划的电视广告与游戏内的活动结合在一起，让用户付费玩游戏产生了很大冲动。

对于日本特殊文化的了解、游戏的美术品质、日本高昂的游戏推广成本和特殊的游戏推广方式，都是中国公司走向日本市场的短板。但中国公司也有自己的优势：一、中国公司的技术实力、技术人才素质已经达到甚至超过了日本本土公司的平均水平；二、日本程序员的工资是中国优秀程序员的好几倍、日本程序员普遍年龄偏大，对于世界最先进技术的跟踪速度相对较慢；三、中国生产的游戏类型丰富、题材多样。因此中国公司扬长避短进军日本市场还是有很大机会的，但必须选择对于中日两国市场和产业都有深入认识的合作伙伴才有可能取得成功。

走出迷局，让用户关系沉淀

陌陌科技运营总监王力专访

记者 / 陈粲然

“每个人都有属于自己的一片森林，迷失的人迷失了，相逢的人会再相逢。”

这是我在使用《陌陌2.0》绑定新浪微博时看到的一句话，它似乎在述说着《陌陌》的现状与愿景：从2011年8月3日，《陌陌》iOS版正式上线一年多来，其产品形式还是相对单一，加上一些媒体先入为主的宣传，导致用户对产品的理解有些偏颇。同时，对于很多用户来说，他们在《陌陌》上将陌生人关系转换成熟人关系后，很容易转向其他社交方式，不再通过《陌陌》联系。所以陌陌科技的当务之急就是帮助用户走出对这款产品的误解，并找到令用户关系沉淀的方式，让“相逢的人会再相逢”。

2012年9月7日，《陌陌》推出2.0公测，我们看到，陌陌科技在这款新产品中引入了新的社交模式——地理位置群组。其产品设计师林志霖在知乎上对群组功能的应用场景进行了描述：“同在一个小区却没机会交流购物生活物业联欢话题的租客们；同在一个城市或学校却没机会聚集互助的留学生们；同在一个公司、一个团队平时聊天扯淡突然想聚会却找不到人的上班族。”

不难看出，群组功能显然背负着沉淀用户的巨大使命。但也有人提出质疑，认为《陌陌》产品本身缺乏熟人社交的基因，加上现在移动端社交工具都具备相似功能，因此陌陌的群组功能很可能成为一个难以建成的“巴别塔”，带着这个疑问我拜访了陌陌科技。

陌陌科技的运营总监王力是其创业初始团队的7位网易离职成员之一，微博ID为“王老板”的他是陌陌成员中最活跃的一位，通过微博把陌陌的运营做得有声有色。而在我与他交谈时才知道，在此之前，他毫无相关经验，因为认为“在大公司内的人生轨迹都是可见的”这个念头，他加入了移动互联网创业大潮，见证了陌陌科技从7人成长到60多人的历程。采访中他从群组功能、盈利等方面分享了对陌陌科技的现状和发展思路。

不一样的群组功能

“我们的群组和他们的是完全不一样的体系”，王力开门见山地表达了陌陌对群组功能的构思：“在米聊等大部分产品中，群组是任何人都可以申请建立的。但我们有很多的规则限制，例如对建群者的要求，必须是使用产品超过一个月，有职业信息并且绑定了个人邮箱的用户，以后可能还会在这块做更细的一些筛选，不是每个人都有权利建群的；同时，我们一般将群组的成员数限制在20，这样不活跃的用户就容易被淘汰，保证用户的沟通，并且是有价值内容；地理位置上做的限制比较多，只能允许你在一个很明确的地点，例如一个街道、写字楼，或者一个小区范围内建立。但如果是一个非常狭窄的兴趣做主题主题，那么我们会允许其范围稍微大一些。为了保证群组的内容，即使群组建立之后，管理者的每次对群相关信息的修改都会被人工审核。”

王力透露，在2.0和今后的小版本中，都会把群组功能作为产品的重心，进行维护和改进。但到3.0时，他们希望打造全新的功能，让用户能够持续的互动，产生一些更丰富、更生动的内容，而不是单纯靠聊天来维系关系，帮助《陌陌》在用户关系中做更深的挖掘。

《陌陌》在开发前没有做过市场调研，创始人唐岩有了这个念头，主要是出于自己的一些经历和判断。在这一年的发展中，运营的节奏主要还是

依靠直觉，数据只是作为一个辅助。不过王力说今后他们会加大对数据的重视，“现在陌陌每天发送的信息量里面，有1/3的信息都是双向关注的好友之间发送的”，这个数据就促使他们会更多地考虑如何将这些已有的关系沉淀和稳定下来。

把钱用在刀刃上

2011年8月时，陌陌科技创始人唐岩仅通过5分钟的面谈就与紫辉投资CEO郑刚敲定了投资。而时隔一年后，陌陌科技完成4000万美元B轮投资融资的传言甚嚣尘上。出于保密协议，王力并没有透露详情，但他分享了自己对于“如何花钱”的想法。

他说这笔钱一方面是要花在对人才的吸引上，在招聘时不用紧紧巴巴的，能够为员工提供一个非常优厚的待遇。CTO李志威告诉我，目前最缺是是iOS开发者，虽然目前iOS平台上开发者众多，但他们希望能招聘到的设计架构思想较丰富，对网络、数据库有一定了解的iOS工程师还比较少。他同时分享了陌陌目前在技术上最大的挑战：《陌陌》一开始使用的是开源协议，但它早已无法满足《陌陌》现在日均4000万消息的量级和峰值100多万的连接数。由于没有太合适的可用平台，所以他们选择了自己采购服务器，使用自主的通信协议。他们需要让服务器顺畅处理每秒几千条的API请求，同时降低通信故障率。

除此之外，在市场投放上的资金也相对比较充裕，在未来可能会有一些比较大的动作。但他表明陌陌科技不是电商，不会有了钱就进行铺天盖地的广告宣传。而是希望以比较精准的方式来进行线上传播。并且，他认为互联网产品的推广一定要基于互联网，绝不会考虑线下的推广模式。

有用户量就不愁营收

在许多人看来，今年的移动互联网创业之路异常凶险。但王力说他觉得目前的形势很不错，无论是市场还是资本，好的产品还是能够脱颖而出的：“巨头的存在确实会对创业者有一些影响，但走垂直细分的道路，还是有很大的机会。《陌陌》要做的就是改变传统互联网的社交模式，建立移动互联网的社交入口，解决有价值用户的社交需求，成为一款有生命力的社交产品。只要迅速地壮大自身，向用户提供有价值的服务，就能稳固地成长，不用惧怕任何威胁。腾讯是个伟大的公司，如果我们对它们构成了威胁，那是对我们的肯定。毕竟有挑战、有竞争，创业才有乐趣，才更加刺激和丰富，否则也太没意思了。”



在创业之前王力毫无任何互联网营销经验

他也认为《陌陌》的盈利模式有很多种可能：“现在门户广告的效果很差，很难引起用户注意，但你想想看，我这么小的一个产品，如果每天能做到门户网站那么多的量，加上在移动端更容易聚焦用户的视点，广告效果要好得多。我们还可以做一些本地化的东西，举个例子，如果你是三里屯的一家商户，我们可以让每个用户到三里屯后，只要刷新了地理信息，你的广告就能显现出来，这种广告模式更为精准……”他说对此他有很多天马行空的想法，但这一切的基础是《陌陌》的用户数到达一个很大的量级，并且有较高的活跃用户。如今《陌陌》的UV值达到了270万左右，王力估算明年这个数值就能超过一些门户网站了：“做社交产品，如果说增长很快，而忽略了用户储备量，用户流失也很快的话，会很被动。因此，未来我们的主要工作一方面是强化用户壁垒；另一方面是让日活跃门户和用户数同步增长。如果这两点能够确保，说明这个产品经得起考验，到那时，我们根本不愁营收。”

移动创业，有梦想就有希望

UC产品总裁何小鹏专访

记者 / 董世晓

2004年，互为大学同学兼亚信同事的何小鹏和梁捷，毅然放弃了光鲜的工作，创办了UC。那之后，何小鹏一直负责产品管理工作，见证了移动互联网的跌宕起伏，积累了丰富的经验。

《程序员》：在当今巨头涌入的移动互联网时代，我们面临着哪些挑战？

何小鹏：挑战可以总结为“四少三多”：“四少”是指屏幕尺寸“少”（实际是说“小”）、带宽“少”（无线传输瓶颈使移动互联网的带宽有限）、输入方式“少”（没有键盘和鼠标）、电池容量“少”（电池容量与体积成正比，无法达到高容量）；“三多”是指终端多、操作系统多、网络环境多。

在这种情况下，UC过去八年最核心的工作就是解决如何在一个低速网络和一个低效设备上，实现移动浏览。在手机不断普及、网络和设备更加复杂的今天，浏览器厂商都不可避免地面临同一个问题——如何在跨网络、跨设备上达到更强的综合能力，使之能够全方位融入到用户的生活中去。

《程序员》：当前有越来越多的App成为内容入口，这会不会挤压浏览器作为内容入口的地位？

何小鹏：这实际上是一个动态的平衡。我们在进行内部分享时，会比较少地关注App对于浏览器的挤占，包括入口时间和入口习惯上的挤占。

从PC互联网到移动互联网，App的形态是广泛且持久存在的。但App用户体验的好坏取决于App的用户数量。App虽在近两年会一定程度上挤占浏览器的入口时间，但它只是一个阶段性的结果。同时，浏览器在增强人们使用习惯方面有天然优势，可以进行应用领域的拓宽。因此，从长远来看，App与浏览器将长存，并随着手机终端设备和网络的提高，浏览器会从App上拿到更多份额。

《程序员》：前段时间与Evernote的合作，是不是你刚谈到的浏览器可以拓宽更多的领域的一种？

何小鹏：普通用户可能会觉得这仅仅是双方在一个功能级别或者在一个扩展特性级别的简单合作，但对我们来说，意义重大。

首先，通过与国际厂商的合作，能够对我们的思路和视野进行一次不同寻常的洗礼。例如，Evernote在iPhone和iPad上实际是同一个App，而并没有像很多应用会分成多个App。它为什么会这样考虑？这对UC产品设计的思路是有启发的。

其次，Evernote在产品方向和全球化上面的布局方法，以及为何它能以百人规模的团队运营一款如此全球化的App，值得我们学习和参考。

因此，以前包括UC在内的很多国内应用厂商，在与国际厂商合作时，更多的可能是在品牌层面，而现在通过合作，我们更应该能够在团队、视野、方法论上去获得互相的学习。这非常重要。

《程序员》：当前，国内移动互联网有没有崛起或者涌现像传统互联网巨头那样的新的平台？

何小鹏：在我看来，截至目前，国内移动互联网尚未有新的巨大平台的出现，因为现有的这个体系内，传统的互联网巨头并没有明确地显现出成为基于移动互联网的新的大平台的可行性。

中国的巨头基本上都不走出去，而且中国用户相对来说，对于隐私、品牌、牵引度等方面与国外用户也大相径庭。这导致的一个严重后果就是——巨头可以通过其渠道去控制用户，甚至更危言耸听点就是它可以将用户玩弄于股掌之上，而且用户还买它的账。因此，这种情况下，移动互联网下一步除非有新技术和新硬件出现，才有可能诞生新的大平台，否则在当前的移动操作系统上很难

涌现像百度、腾讯这样的传统互联网巨头。

《程序员》：走出去也就是全球化了。UC这段时间在走向国际化方面，做了不少工作。对于国际化，你们是怎么考虑的？

何小鹏：几年前，我跟苹果一位主管沟通时，他说了句让我刻骨铭心的话——中国没有一家互联网公司的产品具备全球影响力。虽然同为华人，但他站在苹果的角度上，对中国互联网产品做出的这个评价，值得我们去深思。我们希望能够做成一个具有全球影响力的产品，因此走国际化、自有内核的模式是必须的。

关于国际化动作，我想更多地从产品层面来谈。

首先，可以把国家分为发展中国家、日韩和欧美三类，对于每一类国家采用不同的方法去做产品，用不同的方法进行推广营销。

其次，当移动互联网的产品要走出去时，会发现国际上的终端兼容性和网络兼容性实际上比中国的还复杂。坦诚来说，在我们最开始做浏览器时，有一个判断算是有过失误的。当时我们看国际上的其他浏览器公司，例如Firefox，仿佛并没有做太多事情就解决了终端和网络兼容性问题。结果后来才发现，Firefox在跨国、跨文化、跨网络、跨设备上做了很多，解决了大量本土化和兼容性问题。因此，要想做一个通用型的国际化产品，就需要做大量的本土化，当然，在这里，本土化不代表是运营的本土化，而是技术的本土化，去适配和兼容，以达到更高的可靠性和稳定性。

再次，国外用户的习惯、文化、教育背景等，与国内差异比较大。因此，要根据不同国家的习惯和需求做出一定的变化。例如，中国的视频网站已经相当火爆了，但实际上国外用户看视频的需求比国内还大、频度还高，这是一个非常有趣的现象。再例如，国内用户玩网络游戏很多是按道具收费的，更讲究人员之间的更强、更虚荣、更炫耀，但国外用户却是按软件收费的方式，更多的是一次性付费。

最后，如果说在中国有十种推广方法的话，在不同国家中可能只能选择其中的三到四种，再选择它们本地的几种方法。

《程序员》：除了国际化，UC一路这样创业走过来，肯定有很多创业经验，给我们分享一些？你看到了哪些创业趋势？

何小鹏：创业者没必要一开始就给自己下达指标，要把公司做成多大规模，而是第一步要做出你期望的产品，找到产品的感觉。第二步就是要活着、要能生存。因此，如果要做到这样，团队人不能太多，而是要少而精。

在产品上进行快速迭代，多利用现在的平台力量进行快速尝试和探索，这是对所有移动互联网创业者的忠告。

有较多创业经验或者较好资源的创业者，如果希望将来能成为平台公司或者互联网公司中间的重要一环的话，被收购或者被投资的潜在可能性，在当今的移动互联网还是很大的。我看过一些不错的互联网创业团队，虽然他们的很多想法高得出奇，但我相信这是一个移动创业的好时机，只要你怀揣梦想，就有希望能够实现。

对创业者来说，2012是个好年景，这一年，大家不再浮躁，市场上也不热钱滚滚。这个时候才是真正可以静下心来去做事情，特别对创业者来说，创业成本更低，创业人员会更多。此外，整个移动终端，特别是以Android为主，正在快速增长。对于创业者来说，Android的成长肯定比iPhone好得多。

正基于此，我觉得创业者投身移动互联网创业时，可以考虑三个方面：第一，游戏肯定是一个重要方向，借助HTML5或者Web App，是创业者下一步可以去思考的一种方式；第二，移动企业信息化是一个明显的趋势，值得移动互联网创业者特别关注，特别是当Windows 8推出之后，对移动企业市场会有明显的促进作用；第三，移动生活服务也是一个好方向。



何小鹏认为2012是个静下心来创业的好年景

精益创业和敏捷

文 / 施韵涛

本文结合热销图书《精益创业》中的核心观点，清楚地阐释了精益创业和敏捷之间的内在联系，并在文章结尾得出结论：精益创业和敏捷开发双剑合璧，才是创新型企业成功进阶的利器。

来自硅谷的精益创业热潮

2011年9月，一本题为《精益创业》（The Lean Startup）的书风靡硅谷，不仅仅是硅谷的创业者，很多产品和技术人员，甚至管理层人员也都人手一本，开始琢磨精益创业理念。该书的作者Eric Ries，一位曾经的硅谷创业者，成为了热点人物，被邀请到各种场合，发表关于精益创业的演讲。各种有关精益创业的讲座、课程和工作室（Workshop）也应运而生。不久前有一个“精益创业画布”（Lean Startup Canvas）的活动在上海举办，吸引了众多国内创业人员的参与。

新瓶旧酒还是新兴学说？

在回答这个问题之前。让我们先回顾一下目前常用的产品设计方法。

一个新产品的推出，可能是灵机一动的产物，也可能是大量市场调查的结果，不过能否被市场接受，只有用户说了才算。

有鉴于此，很多软件企业和互联网企业采用了“产品原型”或者“概念验证”的方式。在早期搜集用户的反馈，从某种程度上降低了产品方向性错误的风险。然而，产品原型和概念设计体现的是当时对用户和市场的理解，有些关键需求可能没有在产品原型中体现；在快速变化的市场面前，当初被认可的设计有可能很快就不再适合。

以上这些方法的局限性，使得企业依然要承担很大的产品设计风险。有没有一种方式，能以市场和用户为导向，快速灵活地根据用户需要，适时调整设计方向，从而做出为市场认可的产品呢？有人可能很快回答：敏捷方法。但这样的回答只是部分正确。

我们先来看看精益创业是如何应对这些挑战的。

Eric Ries在他的书中指出，精益创业的关键在于避免因路线错误而造成巨大浪费，通过建立一种开发、评估、学习（Build、Measure、Learn）的文化和机制，快速假设、快速验证、快速调整，有序地进行产品演进和开发。

这里的一个重要前提假设是：我们不知道用户真正需要什么，我们也不清楚适合用户的产品或方案什么。因此，我们来做出一些假设，通过验证这些假设，来找到正确的方向。很显然，我们要作出的第一个假设就是“用户假设”——假设用户有着某种需要。为什么用户会有某种特定的需要呢？这就引出了我们要做的第二个假设——“问题假设”——假定用户遇到了问题（“痛点”）。有了问题，自然会有第三个假设——“方案假设”——假设我们的方案能解决用户的问题。在我们的精益创业课程中，经常会用到一个例子来说明这三种假设的关系（如表1所示）。这个例子是设计一种打扮宠物的工具。

从这个例子可以看出，产品是否有市场，关键在于三大假设是否成立，其中任何一个被推翻，

都将颠覆原有的产品方向。《精益创业》把验证用户假设的过程，称为“客户开发（Customer Development）”和“验证学习（Validated Learning）”。还是以宠物打扮工具为例，说明一下验证假设的方法（如表2所示）。

可以看出，验证假设包含技术的和非技术的方法。最小产品集合（MVP）与产品原型的概念类似，其他市场调查、数据分析及用户访谈等方法，都是高效低成本的有效方式。精益创业主张尽量用非技术方式，经济快速地验证假设，及时作出变化和调整。一次验证一个假设。比如，一旦抽样调查的结果显示大部分宠物主没有给宠物购置衣物和打扮的习惯，就要立刻考虑是否还要继续这个产品。

当用户假设和问题假设得到相当程度的验证时，就可以开始定义并设计开发MVP了。《精益创业》特别强调，MVP不仅存在于产品早期，第一个MVP即使得到用户认可，也不意味着后续的设计和开发就要完全按照第一个MVP的思路走。相反，只有持续不断地通过迭代式演进，持续收集最终用户的反馈，不断调整产品设计、架构、定位和商业模式甚至是销售渠道，才可能最终做出成功的产品。换句话说，MVP是一个系列的产出，在产品的不同阶段，都会有相应的MVP。在整个过程中，不断地调整和变化（Pivot）是关键。

图1中的“数据”指的是各种市场和用户反馈的数据。学习的过程就是根据反馈调整创意和设计的过程。经过验证的学习，就是精益创业的核心理念之一——Validated Learning。

精益创业vs.敏捷——流派之争还是双剑合璧？

敏捷开发以人为核心，以价值为驱动，通过迭代、循序渐进的方式开发软件产品。在开发过程中，强调每个迭代都产生可工作的软件供用户反馈，并及时作出调整。为了做到这一点，敏捷方法通过测试驱动开发（TDD）、持续集成（CI）、结对编程（Pair Programming）、重构（Refactoring）等极限编程（XP）实践来确保高质量软件的及时交付。敏捷方法所倡导的可工作软件、迭代开发

表1 三种假设的关系

假设类型 Hypothesis Type	假设举例 Hypothesis Example	待验证前提举例 Assumption Example
用户假设	我认为用户有为宠物打扮的需要	有的宠物主每个月都要为自己的宠物定制一个小饰品或服饰
问题假设	我认为我的用户在打扮自己宠物过程中不知道该怎么做	宠物主认为准备这些东西很麻烦，浪费时间
方案假设	我认为我的用户会愿意使用我们的工具打扮自己的宠物	宠物主会支付一定的费用来避免这些麻烦

表2 验证假设的方法

假设类型 Hypothesis Type	实验类型 Experiment Type	举例 Example
用户假设	抽样调查；问卷调查；社交网络调查	问卷调查显示有85.2%的用户表示有宠物够来特制衣物的习惯
问题假设	搜索数据分析；网站用户行为分析报告；用户访谈	“宠物衣服布料在哪里买”的关键字搜索结果达到七千万条
方案假设	最小产品集合（Minimal Viable Project, MVP）	有10位客户为所展示的最小产品集合提前付费

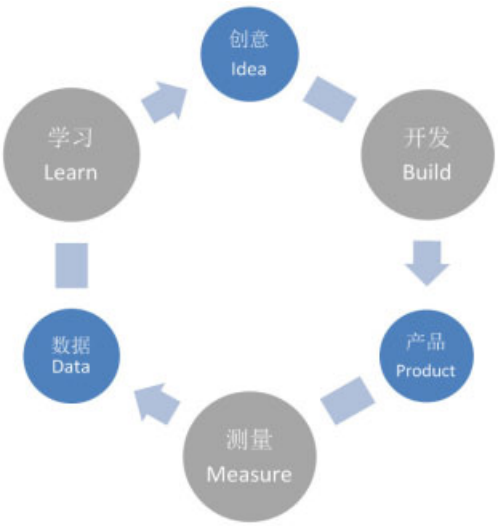


图1 开发、测量、学习的循环过程

和持续反馈的理念与精益创业所追求的MVP、步步为营、不断试错的精神具有很高的契合度。但两者又不尽相同，以下是前面的问题答案“部分正确”的原因。

首先，两者的侧重点不同。精益创业的目标是快速、低成本地验证各种和产品设计相关的假设，避免造成产品上市后无人问津的后果。而敏捷方法的目标是通过迭代式开发，以一系列工程实践（XP）为基础，交付出高质量的软件。

其次，两者要解决的问题也不同。精益创业要回答应该做什么产品的问题，而敏捷要解决的是如何做好产品的问题。精益创业的各种实践，假设、验证、测量、调整等，都是围绕着“什么是用户真正需要的产品”展开的，其中大量的是非技术性的实践；而敏捷方法，除了迭代开发、敏捷

发布计划、回顾等管理实践外，还有大量的工程实践，为的是做出高质量的产品。

从产品的生命周期看，精益创业更侧重前期的客户开发（Customer Development），通过发掘客户（Customer Discovery)和验证客户（Customer Validation），来找到产品准确的市场定位和符合用户要求的设计。敏捷方法通常是在产品定位相对清晰之后，通过1~2周的快速启动（Quick Start），制订出迭代开发计划，然后在开发过程中逐渐完善需求。

既然精益创业和敏捷有这么多的差异，是否能认为精益创业是颠覆敏捷的新方法呢？在精益创业的理念推出后，的确有人提出了“敏捷已经过时，现在进入精益创业时代！”的口号。我认为，敏捷非但没有过时，反而因为精益创业的提出，获得了新的动力。如果说敏捷方法是指导企业如何用正确的方法生产产品，那么精益创业是教大家如何找出正确的产品来生产。精益创业和敏捷结合，就是一个完整的方法论：“如何用正确的方法来生产正确的产品”。敏捷是精益创业的自然延伸，精益创业是敏捷的自然拓展。一个能成功实践精益创业的企业，前提是它已经足够敏捷，否则企业无法落地经过验证的创意，甚至不能有效地小步快跑，完成持续验证的过程。

价值回归——一个现实中精益创业和敏捷开发相结合的故事

REA（www.realestate.com.au）是澳大利亚最大的房地产垂直搜索和广告平台，几年前成功实现了敏捷转型。搜索和相关广告业务的成功，使REA产生了利用长尾客户拓展业务方向的想法。落实这一策略的方向之一，就是建立一个子站，展示各种房屋的内部装修，以吸引原有客户甚至是新客户来浏览精美的房屋装修图片，然后通过页面上的装修和家居广告赢利。经过半年多的努力，新的子站成功上线，但访问量和广告的有效点击低于预期，经过一段时间的运营后仍然没有明显起色，以致管理层产生了砍掉项目的想法。

这时，我们与REA项目组的主要成员，开始通过Workshop的形式，一起探讨产品策略。我们都认为精益创业的思路能解决当下的问题。

我们从网站的定位开始，对原有的各种假设一一进行梳理。例如，项目组原先设想买了房子的客户，应该有装修的需求，但网站的表现并不支持这一假设。当我们把当初的各种假设一一列在白板上，然后检验哪些经过了验证时，发现直到产品上线，很多还是停留在假设阶段，没有过任何的验证，更不用说如何结合REA的现有资源，发挥协同优势。思路明确后，Workshop的一项重要产出物也形成了：假设清单（Hypothesis List）。该清单既有策略层次的假设，例如购买房子的客户在装修时，是否来到这个子站寻求信息；是否有非购房用户直接来到这个子站？清单还包括了战术层次的假设：如用户社区的建立是否能帮助用户更好地作出采购决定。清单也有产品体验细节的假设：如图片分页浏览的体验是否好过图片搜索和过滤？

首批假设清单的建立，为下一步的行动定下了基调。这时，开始采用敏捷开发方式。当一个假设需要开发支持时，开发团队能很快完成高质量的代码，更通过一键式部署将功能很快上线，然后根据网站数据和用户反馈调整设计，再通过开发团队实现。整个设计、开发、部署、验证、调整的过程完美体现了图1所展现的循环，同时将每个循环的周期压缩到了最小，甚至以天而不是周来计算。随着越来越多经过精益创业方式验证的设计被采纳，网站表现开始改善，项目也得到了保留。

仅有好的工程实践不是做出好产品的充分条件，好的创意和验证体系，也不能保证设计都能完美实现，只有创意、验证加上工程实践的紧密结合，才能做出用户认可的好产品。精益创业和敏捷开发的双剑合璧，才是创新型企业成功进阶的利器。



施韵涛

ThoughtWorks中国区客户和交付总监，负责软件交付项目的管理和业务拓展工作。曾任职于德国SAP公司。还曾担任美国TRADOS公司中国区首席代表。

责任编辑：杨爽（yangshuang@csdn.net）

易用性：未来的主要角逐点

又拍云存储CTO沈志华专访

记者 / 杨爽

云存储是云计算领域中的一个重要的产品形态，引得国内多家公司争相进军这一领域，想占有一席之地。又拍云存储是国内较早提供云存储服务的公司，经过了几年的探索和发展，逐渐形成了自己的商业模式。近日，《程序员》杂志采访了又拍云存储CTO沈志华，带大家一起听听国内云存储先行者的心声。

《程序员》：请简单介绍一下又拍云存储的由来。

沈志华：又拍网是我们在2005年的第一个创业项目。经过几年的发展，逐渐成为目前国内较具影响力的照片社区。在这一过程中，图片数量的迅猛增长，致使底层存储无法满足业务发展需求，出现了全国各地用户访问速度不均匀、缩略图作图速度不够快等一系列问题。于是，我们不停地扩容升级再扩容再升级。经过四次比较大的系统升级后，存储平台在文件存储和全国网络分发等方面才逐渐稳定下来。

此时，我们开始考虑，多数网站都不可能一开始就购买大量的设备，由小到大的升级过程几乎是网站都会经历的。既然我们已完善了整个存储平台，那么是否可以将它开放出来，为其他网站提供文件的存储和分发等底层服务呢?

最后经过多次讨论，我们决定将底层服务平台从又拍网独立出来，为其他有文件存储和分发需求的网站提供服务。2009年底，又拍云存储正式从又拍网独立出来。

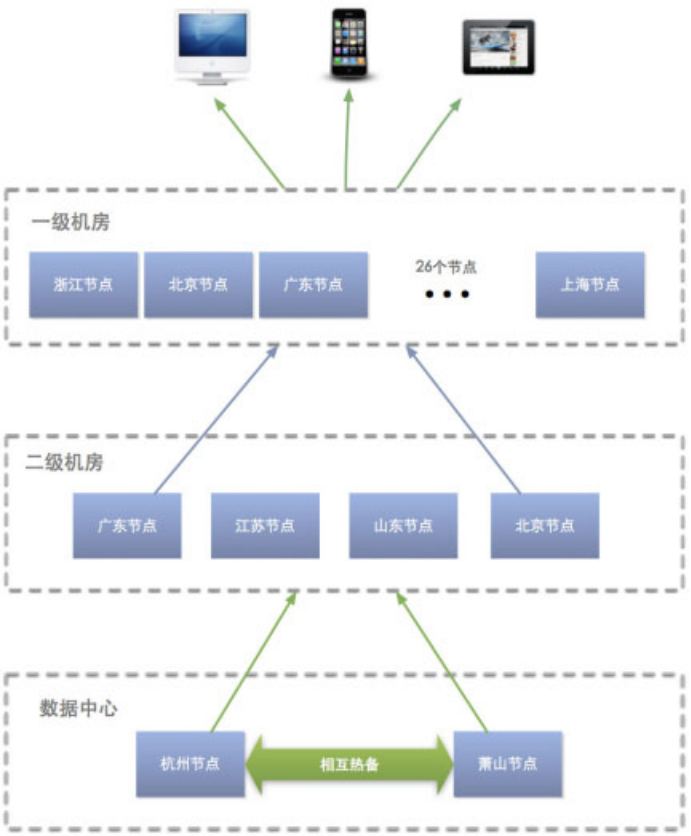


沈志华认为，在未来3年内云存储会成为互联网公司的标配

2011年11月，我们基于又拍网的400多万用户和又拍云存储的成熟底层架构技术推出了花瓣网。又拍云存储的业务是面向企业的，为花瓣网和又拍网提供技术层的支持。

《程序员》：从最初起步到逐渐发展壮大的过程中，遇到了哪些困难？你们是如何应对的？

沈志华：与所有网站一样，又拍网上线后，主要精力都放在了网站的业务层面上，对文件存储分发的研发投入较少，直接买磁盘阵列柜进行存储，买通用的CDN做文件分发。阵列柜满了之后，就买新的，不停地买。这时问题就来了。新增数据都在新买的阵列柜上，新买的阵列柜承担所有的上传压力。这导致压力小的阵列柜很空闲，压力大的阵列柜又太忙，设备性能利用率很不均衡，存取速度变慢。而且每次启用新的设备，业务逻辑也需要进行配置调整，非常不方便。



又拍云存储架构图

因此，我们开始研发存储管理系统，最初用Python开发YPFS，存储端选用MogileFS，解决了文件均衡分布的问题。但由于MogileFS的元数据是集中管理的，所以单套MogileFS能承载的文件数量有限，且随着数量增多，性能开始下降。因此，在运行一段时间后，我们又开始研究新的存储解决方案。

在系统升级的过程中，文件迁移非常麻烦，为了保证业务的正常运行及一些老文件操作的同步性，大约花费了几个月的时间才完成。又拍网发展的最初几年，我们在这些工作上投入了大量的时间，有点儿拖累了又拍网网站业务的发展。

除技术问题外，我们遇到的最大挑战是推广“弹性计费”的概念。2010年初，国内的计费模式是带宽计费，我们是第一家提出流量计费的公司。当时很多人的认知是带宽计费可以预知自己一个月的费用是多少，而流量计费是不可预知的。那段时间，我们几乎是1对1地向数千个用户讲解流量计费的好处，才获得了第一批用户的支持。

《程序员》：又拍云存储的架构实现是什么样的？与同类产品相比，有哪些优势？

沈志华：又拍云存储是一个文件存储＋全国文件分发＋图片集中处理的平台，将云存储的出口布到了最终用户所在省份，避免用户下载时出现过多的省际间时间消耗。我们最开始时用的是通用CDN，但随着文件数量越来越多——有几百TB的源数据，访问又极其分散，如何在各地节点中缓存这么大量的数据，尽量少地刷新数据，需要每个节点有几十TB的缓存能力，这对CDN是个非常大的考验。

另外，为了适应不同客户不同源站的需求，具备通用CDN需要具备较好的兼容性，从而不得不牺牲性能。而又拍云存储只为静态小文件提供服务，可以为了性能弃掉那些不必要的兼容性，并且由于很多防盗链策略无法在通用CDN上实施，因此最终选择了在全国各地布点。

我们的存储系统是参考Amazon Dynamo设计的简单安全分布式系统，以安全稳定为优先，保证每张图片安全存储在3台不同的物理服务器上。

与同类产品相比，我们的优势是系统足够简单稳定，只做100MB以下的小文件分布式存储，可以放弃很多功能及大文件兼容，把小文件的存储分发做到极致。

《程序员》：目前，很多提供云存储服务的公司都在靠提供大存储量或免费空间和服务来抢占市场。这种局势下，你们有什么策略？

沈志华：我们认为针对企业的云存储产品免费是一个错误的决定，至少在当前是一个错误的决定。国内的IDC、CDN都是收费服务，而云存储、云计算服务是更优于传统IDC的服务，所以完全没有免费的理由。

用免费去抢占市场是不明智的，因为云服务是一个大量耗费带宽的事情，其带宽成本非常巨大。免费意味着巨大的亏损，在亏损压力下，服务质量必然会降低。而降低的服务质量会使客户的托管数据访问极度不稳定，这给客户带来的损失是巨大的。

因此，这几年不管多少竞争对手提供免费服务，我们都坚持收费，并且一直坚持为客户提供优质的服务。

我们的市场策略是主攻创业型公司和个人，创业型公司更希望将非核心业务的运维等繁复的工作剥离出来，以有更多的精力专注于核心业务的发展。而且又拍云存储可以让创业者在起步期以最小的硬件投资成本进入，降低了创业的资金压力和风险，这是我们的优势。

而创业者是最好的口碑传播者。通过这几年在创业者中扎根，我们积累了很好的口碑。目前有超过80%的用户都是在朋友介绍下使用又拍云存储的。而这几年我们平台服务的很多创业者都成为了较大的公司，例如周末画报、网创、豌豆荚等。

《程序员》：你刚才提到服务对象主要是创业型公司，那么哪类指标对他们来说是最重要的？

沈志华：之前，猛买网张磊在分享使用又拍云存储的体验时说，有几个问题令创业公司很头疼。

第一，峰值带宽决定大部分成本。尽管大多数时候带宽只有20MB，而高峰带宽为100MB，就不得不买100MB的带宽，这对创业公司来说，成本浪费太大。

第二，运维成本不可忽视。以猛买网为例子，使用又拍云存储之前，开发人员要兼职做运维人员。并且将图片服务和网站服务放在同一台服务器上，互相争用服务器资源和I/O。运维人员需要经常调配置优化系统，花费大量的时间做服务器资源调配工作。

我们可以很好地解决这两个问题。首先，我们是按需付费，就是说你用20MB带宽时就付20MB的钱，用100MB时付100MB的钱。而不至于像带宽付费模式那样，必须为峰值埋单。

另外，这种弹性带宽方式，也便于网站应对突发性的带宽增长。传统的IDC模式会限制带宽的峰值，这样当突发性增长发生时，网站的访问体验通常会下降，甚至无法访问。

前不久，我们看到京东、苏宁大战时，苏宁网站就一度无法访问。而又拍云存储采用的弹性计费方式，本身就不限制带宽峰值，因此完全不用担心突发增长带来问题。实际上一个网站90%的访问压力都是来自静态文件的访问，将这部分内容托管到又拍云存储上后，网站就可以完全解放来自运维方面的压力。我想，这就是创业型公司选择又拍云存储的原因。

《程序员》：一直有种说法是云存储的成本比较高，你怎么看待这个问题？

沈志华：国内大量的网站还都在使用单台或数台服务器，将单点托管于某个机房。而真正的云存储，是存储和CDN兼具的服务，采用的策略是全国布点和分布存储。云存储成本较高的说法多来自这两者的比较。而将用双线机房的成本与云存储相比的话，就会发现云存储的费用会低很多。再加上网站设备的利用率、为网站省下的研发和运维成本，云存储是可以帮助网站节省大量成本的。这也是国外像Zynga等公司都采用Amazon S3的原因。

《程序员》：你怎么看云存储未来的发展？对云存储方面的产品来说，最核心的竞争力是什么？

沈志华：我们认为云存储在未来的3年内，会成为互联网公司的标配。会出现两种形态的云存储：一种是以支撑自有业务生态系统形成的云，比如腾讯为将自己的面向消费者的业务串联，以支撑开放平台上众多的服务商，一定会发展壮大自己的云存储；另一种是第三方云存储，除腾讯等超大型公司外的互联网公司提供存储服务。又拍云存储力争在第二种形态的云存储中保持优势。

对于云存储产品来说最核心的四个要素是：安全、稳定、快速和易用。而安全、稳定、快速在未来可能会拉平，但在易用性上，将是一个更为个性化创新的领域，这将是未来云存储主要的角逐点。各家能否让自己的产品不断地适应用户新增的需求，能否让用户的使用成本越来越低，将成为云存储服务的核心竞争力。



Marty Cagan

过去20多年，Marty Cagan作为负责定义和开发产品的高级经理人为多家一流企业工作过，他亲历了个人电脑、互联网、电子商务的起落沉浮，致力于通过写作、演讲、培训帮助客户打造富有创意的产品。

产品营销

文 / Marty Cagan　译 / 孙洋

Marty Cagan是享有世界声誉的产品管理专家，曾经担任网景副总裁、eBay产品管理及设计高级副总裁。本文是他与合伙人回顾多年来从事软件产品营销工作的总结和经验分享，分析了产品营销的价值，并分享了文案写作的经验。

产品营销的价值

简是一名市场营销人员，在公司的安排下参与产品X的新版本发布，这件事情公司非常重视。于是，简带着发布清单，召集产品经理约翰开了个碰头会。约翰回答了她的所有问题，在白板上写下了产品的市场细分和产品的核心卖点。简做了大量笔记，然后发了一封邮件给约翰，要求他对邮件内容进行评审。

约翰得到的是一篇新闻发布稿。产品特性混杂不清；没有任何市场定位，仅仅是宣布这个产品“现在已经上线”（now available）；客户推荐语写得死板生硬，仿佛就是Web 2.0机器人直接抓取的。约翰做了大量校正工作，几乎重写了一遍。

后来，约翰从网站上下载了一份产品描述，发现他修改的新版本特性基本没写上去。约翰稍微有点儿担心，简和市场营销的同事没有真正理解他的产品。

约翰找到了市场营销部门的老大鲍勃：“我们难道还用老产品的介绍进行新版本发布吗？”鲍勃检查了发布清单，所有的事情都列在计划表上。他向约翰担保：“你放心，一切进展顺利。”约翰认为鲍勃的意思是：产品营销只是整体市场营销宏伟战略的一部分而已，它们之间唯一的关联是市场需要听到的声音，细节没那么重要。

果然，约翰接到了简的第二轮会议邀请，要求重新“评审产品市场定位”。简在会议上问道：“现在可以告诉我，你希望如何定位产品X，需要我们宣传产品的哪些特性？”

你是否对这种困惑和沮丧的场景感到似曾相识？没错……你并不孤单！

问题

产品营销与企业市场营销的最大区别在于：产品营销只聚焦于特定的产品，这些产品是大多数技术导向型公司的最大资产，正是它们驱动了市场战略以及市场份额的增长。当公司处于创业阶段、只有一款主打产品时，“市场营销”和“产品营销”没有任何区别。之所以演变为两门独立学科，是因为公司产生了多条产品线，或者走专业高新技术路线，又或者是公司部门变得更复杂。

职业

关于两种角色的系统划分，我还没有发现哪一家公司比微软做得更好，我的职业生涯也是从微软开始的。他们认识到这两种工作通常需要不同的职业技能和不同的人才能做得好。产品经理（从项目经理职位演化而来）定义即将

开发的产品。产品营销（从产品经理职位演化而来，职能较混乱）则确保全世界关注这个产品及为什么关注。

正是“为什么关注”这个部分，需要更多的产品知识和沟通技巧。要磨炼这种令人信服的战略表达和信息传达，需要对客户有很深的理解及善于分析的头脑，以及对整个行业和商业的广泛涉猎，还有——最最重要的——真正理解产品中的各种“为什么”。

产品营销的职责

合格的产品营销人应该掌握以下技能。

- 竞争对手分析（竞争对手正在说什么，市场反馈将会如何）。
- 市场和消费者研究（这一点有些超出本文的范畴，它是一个契约模型，关于人们是如何影响产品转化以及人们对产品的感知）。
- 产品定位（产品是什么？为什么它与众不同？为什么人们要关注它？）。
- 产品沟通/公共关系/社会化媒体（正是这些，使人们如何谈论产品）。
- 市场营销活动、广告、E-mail、互动、搜索引擎、事件营销等（选择某种营销手段，就要坚持下去，直到达到极致）。
- 销售支持、销售工具（包括用产品演示及所需的产品知识，创建一个有说服力的销售模型）。
- 垂直市场/分销/传销方案。
- 可量化的营销方案（ROI以及商业目标分解）。

在消费品公司中，产品营销由市场活动驱动，注重产品的购买、使用、保持和推荐。

在大企业中，产品营销注重产品的感知和塑造产品的竞争领域及销售渠道。

产品营销工作，填补了产品和企业营销队伍间的空白，确保准确传达新产品的信息和营销策略。

怎么做？

从产品营销的角度来看待产品计划的必要变更。在整个过程中，让产品营销人员参与产品计划的讨论，确保每一个产品的努力都换来了最大的市场回报。**请记住，如果人们没听说过一款产品，或不知道它的重要性或并不关注它，那么即使产品再有价值、可用性再好，也是白忙活儿。**

即使在拥有产品营销的公司，由于技能的差异，或是对远景缺乏同样的理解，产品经理和产品营销同行的关系也会变得越来越边缘化。不要让旧习惯阻碍了角色定位的改进，确保产品经理和产品营销同事保持沟通，彼此都能收获更多的价值以及加深对市场战略的理解。

相互适应，让产品走上正轨，每一家公司和团体都必须找到一个平衡点，使之为团队力量做出贡献。最理想的情况是，产品经理觉得产品营销伙伴是团队里重要的组成部分。同样，产品营销人员通过产品经理来衡量自己工作的价值。

无论如何，要确保全世界都知道为什么你的产品如此重要，并投身于优秀的产品营销！

为什么传统文案会失效

多年前，我还在微软做Office的产品经理时，曾花费几十万美元进行产品定位研究。产品广告会长期投放在固定的地方，文案“好不好”就显得非常重要。因此，从文案撰写人手里接过写好的文案后，我们还会字斟句酌一番。

可如今，这些产品文案统统失效了。为什么会这样？因为除非你有上千万的营销预算，否则你的文案是无法精准地触及用户的。

如今，文案的作用不再只是简单地传递信息。文案需要干货，要简短、易于理解和接受。产品特点、用户案例、充满热情的客户推荐、评价、详细说明等，都是文案的重要组成部分，它们让客户感觉产品更加真实可靠。现在，只用几行文字描述产品特性的文案完全不管用了。

文案中常会见到以下三个问题。

一厢情愿地认为客户会关注我们的新产品。作为技术从业者，我们总以为自己的产品很新潮，而用户都喜欢尝鲜，因此产品绝对能吸引住用户。但实际上，人们选择产品的的原因是：“我喜欢它，它解决了我的问题，太棒了！”

不考虑文案的生命周期。所有文案都有生命周

期，客户在接触文案的过程中，分为吸引、理解、记忆、推荐和购买等阶段。也许一个好的创意对某一个环节有效，而对另一个则无能为力。绝大部分文案都在针对其中一个环节下功夫（通常都是吸引），而不是针对过程中的不同环节，采取不同的方案。因此，对文案传播的整个生命周期进行过程分步测试，才能有效解决这个问题。

文案与用户接纳曲线不匹配。除了文案生命周期，还应该考虑到早期用户和主流用户需要从文案中获取的信息不同。这也是文案出现多样化的原因。文案必须对所有用户适用，在特定的时间让特定的听众听到他们需要听到的声音，保持品牌和产品定位的连续性。

在检查文案时，以下几点需要思考（本文不探讨文案的设计和展示）。

清晰胜过完整。按照“传统”文案标准，37signals的文案相当不错：“立即下单，没有条款约束，没有隐性收费，没有任何意外。”（Pay as you go. No long term contracts. No hidden fees. No surprises.）它把大量信息浓缩在短语中，不仅容易阅读，而且有一种诗歌般的优美。但现代文案还应该进一步精简，使用更清晰的措辞，例如：“注册只需60秒，选择一个项目即刻出发！”（Sign-up takes less than 60 seconds. Pick a plan to get started!）文案的信息量不宜过多，要让用户的注意力时刻保持在需要做的事情上。

真实胜过权威。研究证明，最容易理解的文字形式，莫过于洁简的短句。有许多网站仍在使用这种格式。但它也存在一些副作用，如不能突出重点或强调语气。我们对比一下Expedia和Kayak的宣传语（Expedia和Kayak是两家旅行网站）。

为何在Expedia订购：
机票、酒店预定，天天有低价，时时有惊喜。
无需机票预订费。
更多选择，更多欢笑就在Expedia……

这文案很枯燥，很容易被忽略。大家也有同样的感受吧。让我们来看看Kayak的对话式文案：

Kayak，您与众不同的选择。
在Kayak，您可以对上百家旅游网站进行一键比价（对的！您不需找20家旅游网站反复搜索）……
亲，Kayak完全免费哦。

Kayak的文案传递了大量品牌个性信息，这些让Kayak显得与众不同。

评论式文案。从接收信息到采取行动，不同的人需要的刺激是不一样的。但列举事实比单纯的推销会更有说服力。Mint.com在这方面就做得很棒，它们在主页上直接提供大量案例，来说服人们使用它们的服务：口碑、主流媒体推荐、产品操作视频（大部分产品视频仅仅告诉你该产品是什么，而不是怎么用）、产品能解决人们什么问题，以及它如何控制风险。口碑营销最近很火，但一定要展示用户的真实评价，切忌推销用语。别让用户辛苦地寻找购买的理由，要积极主动地展示产品的价值。

测试。使用网站分析系统和A/B测试，哪个文案最好，“让我们用数据说话！”不要套用别人的文案。有些文案很棒，但不一定适合你的产品。记住，在“正式”文案出炉前，文案需要不断改进。

你认为以下两份文案哪一个转化率更高？

A.“获取免费邮件更新：已有14,752人加入。”
B.“获取邮件更新（免费）。”

B版本的文案转化率超过A版本的一倍！这与大部分人的直觉并不吻合，特别是口碑营销如此火爆的今天。但对该网站来说，在文案上展示口碑数据没有任何意义。因此，不要做任何主观假设，持续测试一切可以测试的内容。（更多资料请参考：http://diythemes.com/thesis/increase-conversions-split-testing/。）

希望本文能够让你通过一个崭新的视角，来认识产品营销的价值，并撰写更具影响力的文案。请记住：用客户希望听到的语言来吸引他们，并测试文案内容。这样才能创作出一份真正能与用户进行心灵沟通的文案。



本文节选自华中科技大学出版社《启示录：打造用户喜爱的产品》一书及作者的博客。该书从人员、流程、产品三个角度介绍了现代软件（互联网）产品管理的实践经验和理念。特此感谢华中科技大学出版社与Marty Cagan先生授权。

一分钟先生

如何有效地控制项目进度

能否有效地控制项目进度是决定项目成败的关键，也是项目经理工作能力的重要体现。本期三位嘉宾将结合自己多年的实践经验，与大家共同探讨有效控制项目进度的技巧。



白天
北京合辉信息技术有限公司CEO

控制项目进度的几点技巧

软件开发的项目周期大体分为3个阶段：获取需求和定义产品、开发和测试、部署和运维。

■ 在获取需求和定义产品阶段，需要防止的不是进度太慢而是过快、过草率。特别是对于创业公司的产品经理来说，很可能因为看到开发人员无事可做而感到压力，所以尽快完成产品定义，而没有充分了解市场和竞争对手信息，没有与合作伙伴充分沟通，没有做深入的思考。这些因仓促而隐藏的问题，发现得早则导致开发阶段大量返工，发现得晚则导致产品上线后不受欢迎。常听一些人说现在互联网开发，讲究快速迭代和敏捷，边做边想，返工也正常。这是一个误解。快速迭代指的是将不同版本之间的周期缩短，小步快跑，而不是在一个版本的周期内来回折腾。

■ 在开发和测试阶段，项目管理重在跟踪进度和保持沟通——用集成和演示跟踪进度，基于Bug沟通问题。

要做到各个模块外部接口相对清晰稳定，并尽早完成各个模块间的集成，最晚不超过开发周期的1/4时间。第一次集成之后，就应该开始每日集成和每周演示。每日集成使得测试团队每天能同步测试最新的代码，帮助开发团队尽早发现问题并及时了解技术细节上的进度；每周演示使产品经理、项目经理和管理层能从用户的角度感受产品，使他们对产品有信心。集成和演示是项目管理的心跳，合理利用它们，有助于及时把握项目的健康程度。

无论开发流程多敏捷，工程师能力多强，记录和跟踪Bug都是必不可少的。开发团队和测试团队的沟通都应该基于Bug，才能言之有物。开发工程师每次提交代码都应该记录是针对哪个Bug的，每日工作简报都应该写今天关/开了哪些Bug。要在每日晨会（站着开，一般15分钟内）时说好，今天打算解决哪些Bug，其中有哪些点不清楚，需要和谁沟通。

■ 在后期部署和维护阶段，要快速响应。考验的是团队成员的责任心和抗压能力。系统运维工程师要深夜工作，因为部署可能要在流量低的时候进行；项目经理要保持能随时沟通，做出快速而准确的决定，鼓励团队并做出表率；一旦出现高危害Bug，开发团队要在24小时内准备好补丁。Amazon的做法比较有趣：在产品刚上线一段时间内，开发工程师要保持24小时开机。如果自己负责的模块中出现高危害Bug，那么很可能会在深夜被系统运维工程师叫醒。这样不仅能保证快速响应，还能让工程师意识到：前期代码不好好写，后期就别指望能好好睡觉了。



浦薇娜
天猫（淘宝商城）核心系统负责人

项目延期原因及应对之道

每个项目经理都希望能有效地控制项目进度。但这件看似简单的事情，实际操作起来却常常不尽如人意。即使在成熟的大公司里，有着完善的项目管理流程，配备着一流的团队，项目延期事件还是频频发生。这里分析主要的三个原因。

常见的原因之计划不清

很多项目经理，项目完成得很好，计划也做得很漂亮，却总是计划赶不上变化。原因在于，有些时候，按工作量预估的发布日期却得不到领导的同意，领导有时会说我们现在就是和时间赛跑，这个项目必须在某某时间发布。这将致使计划推倒重来，一切都要赶进度。而对于其他团队成员来说，这份计划没有同他们商量，无异于强压任务。项目还没开始，抱怨声就不绝于耳。因此，项目工具选得好、任务划分细致清楚只是做好技术的基础，更重要的是项目计划要得领导和团队成员的认同，并愿意为之全力以赴。

总之，想做好项目计划，要做好以下三点。

- 项目计划前，先和产品经理、上级领导沟通好，确定这个项目的轻重缓急。
- 团队成员要达成一致意见，项目经理不可独断专行。
- 项目计划要细化到天、功能点要责任到人、确定里程碑点。

常见的原因之需求问题

需求中的功能点要在PRD（产品需求文档）中罗列清楚，业务流程要写得完整清晰，交互细节要体现在视觉稿中。要组织项目组所有成员参加PRD评审，评审时要针对具体的问题，给出明确的处理意见。暂时不能确认的问题，问题跟进人要在限定时间内给出反馈，项目经理可以制定问题跟进表格。

项目进行中的需求变更，尽量在前期提出。在项目管理的过程中，当前期的需求和计划都确定后，项目经理不能只顾着跟进开发和测试的进度，也要阶段性地和需求方多沟通，让他们及时反馈意见。不要等到临发布时，产品经理跑过来说“我要的不是这样的，这里要改一下”。永远不要把问题留到最后一分钟，要超前一步，留有余地。下面是一个真实的案例。

案例情景：该项目的整个周期为2个月，有3轮功能测试。当第3轮功能测试结束时，也就是即将进入预发布阶段时，产品经理才给出用户反馈并要求按用户的反馈修改。改动的地方涉及到页面的样式、文案、SQL语句和校验逻辑等，总共可能有20多个文件要被改动。

项目经理建议只改页面的样式和文案，其他部分先不要改，等下次升级维护时再改，否则可能会影响发布。而在多次交涉无果的情况下，开发人员只能硬着头皮修改，测试人员只能再重新测一轮。虽然大家努力地按需求方的要求做了，但项目延期已不可避免了。

常见的原因之沟通不畅

为某项目临时组建的团队往往来自不同部门，团队成员之间不熟悉，此时，要为团队建立一个沟通通道，确保沟通顺畅。常用方式为：

- 建立一个内部网络空间，所有文档资源统一存放，供团队成员共享；
- 利用即时聊天工具，建立一个项目群，每天通报项目进度；
- 建立项目邮件组，所有变更达成一致后，发送邮件确认；
- 每天要开15分钟晨会，每周一次周会，每周发送项目周报；
- 跨团队项目，最好申请独立的项目室，所有项目组成员坐在一起工作，降低沟通成本。



李晨旭
方顺攀藤（北京）网络科技有限公司
研发中心经理

重视细节

项目管理的目的是能够按照预定的成本、进度和质量要求顺利地对人员、产品、过程和项目进行分析和管理。在项目管理中，有些细节需要引起项目经理的重视。

根据经验规划

即先做少量的规划，再根据实践过程中得到的信息来做进一步的规划，这样可提高项目的可行性。试图预测未来的规划很难奏效，除非你是个预言家，否则应该尽量在项目中根据经验做规划和日程安排。

安排项目日程

首先，要按可交付物安排日程，而不是按功能；其次，要以迭代的方式安排日程；再次，要使用难度较低的工具安排项目日程。过度追求完美的项目时间表可能意味着在实际项目中浪费更多的时间。

足够的时间规划

日程安排是由整个项目团队共同制定的，因此，每个人都要对日程有信心。不过，天有不测风云，总会发生点儿意外，所以我们要做足够的时间规划，而且要使用波浪式规划，这样才可以随着环境的变化灵活地更新日程安排。

管理会议

在组织项目时，项目经理要尽量避免浪费时间的会议。要让团队将注意力集中在项目上，这是最简单、最有效的方式。在帮助团队朝着合理的交付截止日期前进时，要保证团队不受外界干扰和影响。如果会议对于任何人都毫无价值，那就取消掉；同时准许团队成员不参与无法贡献和收获价值的会议。也许有些团队成员会不高兴，认为你觉得他们不够重要从而不能参加会议。要跟他们解释清楚，你不让他们参会是因为他们太重要了。

速度图

如果只能绘制一个图表，那就选择速度图。速度图集三种度量方式为一身：需求、已完成工作和时间。虽然无法从中看到自己希望了解的缺陷率或成本，却能从该图中对项目的整体进度有所掌握。使用速度图可以使你在一张图中同时度量多个趋势：整体需求数量和已完成工作，其中包括所有的测试、文档以及项目需要等其他内容。这是最有用的图表，是项目经理的好朋友。但要注意，速度图只是获取数据的工具，不是目的。

测试

从项目开始就要坚持“减少技术债务”的原则，让测试与开发同步进行。测试会将项目的风险展现在众人面前，大家越早看到这些风险越好。在采纳顺序式生命周期的项目中，要让测试人员参与到需求分析阶段，询问他们关于产品需求的反馈；在采用迭代式生命周期的项目中，要请测试人员帮助评估原型；使用增量式生命周期的项目，只要有可供测试的部分，就可以让测试人员尽早开始测试功能；在实施敏捷的项目中，要确保测试人员与开发人员一起工作，以开展技术层面的测试。同时，还要让测试人员与产品负责人一起，编写面向客户层面的测试。

移动产品用户体验与设计系列（四）

简单的设计

文 / 林敏

什么是简单？很多时候我们把简单和简洁混淆成一个概念，以为简单就是使用尽量少的视觉元素。其实不然。简洁和简单关注的角度不同。简洁强调的是“洁”，是干净；而简单强调的是“单”，是不复杂、容易——容易获取、容易理解、容易使用。

Google的主页很简洁，但并不简单。为人诟病的“更多”的设计，以及功能极其强大的高级搜索语法都在向用户说“其实我并不简单”。随着Google的成功，不少搜索引擎都乐衷于仿效其设计，却往往只是套上相似的“简洁不简单”的外壳，而没能在简单上下功夫，做到比Google更胜一筹。

反过来看，网址站的设计不论放在哪个设计师眼里都会被鄙视。但就是这样毫不简洁的设计，却是一个简单的设计。使用网址站的用户，既不需要有多高的学历（识字就行），也不需要有多深的计算机知识（会用鼠标就行）。当年还在美国求学时，系里的教授时常看着新浪主页百思不得其解，流行的解释是东西方文化差异。其实没这么玄乎，当年的门户不就是扮演着网址站的角色吗？要让广大教育程度相差悬殊的网民都能使用，网站的设计就不得不“向下兼容”，简单原则重于简洁原则。

今天既简洁又简单的设计当属苹果。通过简洁的外观加上简单的界面，苹果塑造了许多设计经典。而在功能机时代，诺基亚则是当之无愧的简洁又简单的设计表率。可惜时过境迁，触屏改变了过去的交互模式，让诺基亚曾经的简单变得不再简单。

那么，如何做简单的设计呢？

首先，是做到用户价值上的简单。在核心用户价值上只保留最重要的一项，其他都可以先舍去。这样做最大的好处是用户能够很容易理解你所要表达的价值。就好像谈到搜索引擎用户就很自然地想到Google。

其次，是做到心理上的简单。视觉是很重要的手段之一。适当地借助视觉通过良好的组织让设计第一眼看上去便显得简单。这是简洁可以帮上忙的地方。不只是设计师容易把简洁当简单，用户更是如此。不过，也不能因此就处处以简洁来包装。有时，不需要简洁才是更好的简单，就如网址站的陈列方式。

再次，是做到使用上的简单。这主要是与可用性相关的内容，例如减少步骤、保持一致性等。使用上的简单要根据用户人群的特点来考虑不同的设计策略。同样是命令行形式的交互手段，对于普通用户而言可能非常复杂，而对于软件开发人员而言则是简单的。使用上的简单也要伴随着技术的发展而不断进步。Google地图街景的出现让数字世界变得真实亲切。

最后，是做到文字上的简单。图形界面听上去是图形为主，但不可忽视文字的重要性。一个较为普遍的例子是现在许多公司都非常用心地对404页面加以设计，通过简单而有意义的文字将错误信息传递给用户。虽然排在最后一点，但文字的重要性丝毫不比前三点差。

这，就是简单的设计。



林敏

用户体验践行者和布道者，曾创立UU网为国内早期从业人员提供学习交流平台。现为三星中国设计研究所用户体验创新部负责人。

责任编辑：陈博（chenbo@csdn.net）

手机游戏付费模式分析

文 / 郑金条

移动游戏无疑是现阶段最赚钱的应用类型，但其盈利模式可不止一种。本文对当前的收益模式进行划分，详细探讨了在不同的情形下应该选择怎样的付费模式。

我仔细比对了App Store和Google Play游戏的营收渠道，大致可以这样划分：

- Paid，玩家需要对游戏进行优劣判断并预先付费才能实现下载进程；
- Lite+Paid，玩家可以事先体验游戏的简版，再根据体验情况执行完整版付费；
- Paid+IAP，在付费游戏中为了平衡游戏难度系数而植入具有特殊性能的道具；
- Paid+IAD，付费游戏，但为了推广关联游戏或者广告而植入了品牌广告展示位置；
- Free+IAP，免费模式，依赖游戏中的道具系统收费；
- Free+IAD，免费模式，依赖游戏中的广告系统收费，又分为两个层面：一般的广告系统植入或者广告主品牌植入；
- Free+IAP+IAD，免费模式，执行道具付费和广告展示收费两个渠道；
- Pay What You Want，按照玩家的意愿付费，一般为Humble Indie Bundle模式，目前在手机游戏领域暂时运用比较少，典型的也就是Binary Hammer开发的时间管理工具；
- 分级订阅付费，按照拓展版本分批付费，最为典型的应该是Square Enix旗下的手机游戏《Final Fantasy Dimensions》，共分为四章，每一章都有自己的独立售价，总价为19.99英镑，但分批订阅需要1.99英镑+3×6.99英镑；
- 包月付费，可分为两个层面：针对特定游戏下载批量包月，只要在下载包范围内就不再做限制或者针对游戏道具消费包月，主要是提供更具性价比的优惠道具包，包括允许以更低价格消费包月之外的道具；
- 特殊拓展付费，这个和分级订阅有些差异，有些主要是针对游戏中高级的竞技场或者锦标赛进行特殊的付费设定；
- 通过玩家交易抽成付费，允许玩家进行游戏内交易并抽取比例费用，诸如My Vineyard推出的《General Store》或者国内的《征途2》游戏；
- 通过游戏周边实现营收，例如玩具、T恤、箱包、图书和影视作品，这个在目前游戏品牌化的浪潮中表现得更为明显，典型的诸如：Rovio或者Outfit7；
- 通过玩家捐赠收费，这个和Pay What You Want有些相似，但各自的表述需求完全不同，以捐赠名义出现的更多体现了公益姿态，这类游戏不预设任何付费模式，由玩家自愿捐赠。

	Paid		Free		Grossing	
1	Ultra Contacts Pro for Mass SMS/EMail	=	$战魂	=	$龙之力量	=
2	$Super Sprites	▲2	$捕鱼达人2	=	$神仙道HD	=
3	$Fruit Ninja	▲4	CarrotFantasy	▲1	$雄霸天地(简体版)	=
4	$宠物猎人G	▲1	云和绵羊的故事免费版	▲22	$忘仙HD	=
5	$植物大战僵尸	▲4	银河创世录	▲16	$三国来了	=
6	$Asphalt 7: Heat	▲5	好大夫在线	▲1	$逆轉三國	=
7	$Where's My Water?	▲5	$人人怒海争锋	▲6	$博雅德州扑克	=
8	Mapbar Navi	▲22	吴邪的私家笔记	▲23	$水晶三国	▲50
9	$家园7：贵族	▲4	$乱世天下：南蛮入侵	=	$调教三国	=
10	WeicoPro 微博客户端	=	唱吧 - 你的手机KTV	▼5	$三国志ONLINE(中国版)	▼2

中国区营收榜中游戏占据绝对优势

Free+IAP和Free+IAD

事实上，我们在分析手机游戏营收核心渠道时，特别是在对照国内模式的情况下，Free+IAP和Free+IAD基本是绝对的主流模式，而如果需要在两者之间决出胜负，基于营收规模的考量，Free+IAP无异于是具有天然优势的。一部分原本依赖Casual游戏累积了大量活跃用户的开发者也不再满足于Free+IAD的营收水准，迎身介入更具效益拓展性的Free+IAP。

尽管Flurry在分析手机应用市场效益时认为广告在App市场的营收份额从2011年的总占比18%提升到23%（其他的营收方式下滑为77%），且移动广告的终极效能正在被市场逐步认可（我曾解析过这些有利的因素，包括手机便携性，使广告展示具有随时化可能性；基于LBS使广告具有本地化定位效能；通过二维码和语音检索串联现实广告和手机世界；Social化，体现受众参与意图；即时化，更易形成冲动消费）。

但手机游戏广告终究有不少瓶颈式障碍，包括：

- 如何平衡收集用户行为数据和保护用户隐私？对用户权限的渗透在推动广告更为精准化之外也将行业带向被用户拒绝的风险之下，例如Google Play最新的开发者政策就要求必须明确应用中的广告或通知信息的来源或者允许用户直接越过广告请求；
- 在界面展示有限的情况下是不是对用户的体验产生了干扰？尽管广告业正在优化广告展示的方式，特别是匹配性和互动性层面；
- 是不是存在恶意和无效的广告点击打压了广告主在移动领域的投放信心？按照Trademob的调研，这个数据一直居高不下，和PC的作弊基本没有什么差别，40%的广告点击转化率低于0.1%；
- 是不是因为广告展示行为增加了用户的耗电行为？例如开发者用于生成应用内置广告的第三方代码效能偏低频繁损耗用户的手机电池。

更为极端的是手机应用广告目前并不在其他的App获得下载的核心渠道范畴，按照Ampulum和BI Intelligence从不同的受众角度关于应用下载获得渠道的研究得到的结果都显示App最佳的传播渠道是口碑传播（以品质和机制为契点，基于用户的认同感和喜悦的相互分享，进而渗透到开发者关系、用户关系和媒介关系），其后是应用商店的推荐、应用商店的检索以及外部新闻评述，再之后才是游戏内的广告推荐，Torsten Reil（Natural Motion）就认为传统营销方式对现在App下载市场并不奏效（代之的是检索、展示以及口碑或者信息分享）。

在这个角度上，IAD模式就是偏于传统的方式，如果说IAP模式可能干扰的是用户更为深度的游戏体验的话，IAD就可能很直白地影响了用户对界面操控的体验。

因此，我更倾向于IAP模式的方式。BI Intelligence关于IAP的分析提供了有力的佐证，在关于iPhone App Store营收榜前100的分析中得出的结论是66%的应用采用的是免费模式（相比之下，采用付费模式的应用仅为34%），并且这100款应用（绝大部分是游戏）中高达93%采用了IAP付费模式，仅有7%没有采用IAP付费模式。

当然在这个趋势下，将产品转化为Free已不再是

颠覆营销模式的创举，而是浸入市场获得基础活跃用户的标配，这就必然产生另外一个全新的问题：在全体游戏向免费靠拢的现在，用户规模是不是游戏获得营收的必要条件？当然我们知道越庞大的用户群对于开发者的品牌化或者效益套现或者风险市场融资更为有利，而为了更多的下载量，部分经典游戏推出LITE版，例如《Angry Birds Space Free》；更有直接转向免费的，例如《Cut the Rope: Experiments》、《Jetpack Joyride》或《Temple Run》。

因为按照Xyologic关于App Store/Google Play美国Top Free榜的研究显示，从2011年4月到2012年4月之间，新游戏想要进入Top Free 100的可能性已在变少，以App Store为例，2011年4月当月进入Top Free 100为422款，到了2012年4月当月进入Top Free 100就减为277款，这也意味着新晋应用挤入榜单变得更加困难；我们可以从另外一个角度来看待这种进榜的难度——当月发行的应用能够进入Top Free 100的概率。2011年4月的入榜概率为0.89%，而2012年4月的入榜概率降低为0.28%（前者为后者的3.2倍）。

这时来探讨在无法挤入下载量Top 10，用户绝对基数不高的情况下如何同样具有超级营收能力就显得异常关键。可以参看Flurry对应用营收中长尾的分析，从2010年到2012年App Store和Google Play应用营收与排名关联度的比照。在2010年，Top 25营收占比28%，26到100名占比27%而100名以外营收占45%；到了2012年，Top 25营收占比15%，26–100名占比17%，反倒100名以外营收占68%，这就意味着大量有效下载并不出彩（例如百万级）的游戏凭借着超高ARPU值也同样有不俗的营收表现，诸如Double Forte在4月份提到的Kixeye旗下游戏ARPDAU为0.8美元、TechCrunch在6月份提到的DeNA旗下游戏《Rage of Bahamut》的ARPDAU为0.6～1美元。

其他的收费模式情况

除了Paid、Free+IAP和Free+IAD之外，其他的模式尽管也以各种形式存在，但事实上它们被运用的比例并不算太高，诸如Pay What You Want在手机应用领域就比较难找到太多的案例，典型的也就是Binary Hammer开发的时间管理工具；分级订阅付费（最为典型的应该是Square Enix旗下的手机游戏Final Fantasy Dimensions，共分为四章，每一章都有自己的独立售价，总价为19.99英镑，但分批订阅需要1.99英镑+3×6.99英镑）也更多仅限于超级大型的游戏。

目前运用得更多的，差不多则是包月付费、周边营收以及Lite+Paid模式。

Lite+Paid模式

Lite版在App Store前期是被付费游戏运用最娴熟的用户捕获工具之一（当然还有限免也是很强的用户吸纳利器，特别是专门为用户提供限免服务的App包，不过这种方式目前遭遇了苹果App Store的打压，预计在有影响App Store排名因素的猜忌下将有可能被处罚），提供给玩家免费试玩的可能（当然很多只是从头开始体验，在一定阶段后嘎然而止，需要用户付费才能继续体验）。

一般来讲游戏体验简版（Lite）遵循的是循循善诱让用户自觉迷上游戏并购买的原则，但事实上目前在任何应用商店玩家的可选择性绝对超过想象（数以十万计的游戏选项），不第一时间抓住玩家就意味着错过一个玩家，并且如果在玩家准备入戏就终止强制付费选项就必然导致逆反的心理，可以看到大量的Lite版其实有效的付费转化



《Final Fantasy Dimensions》是市场上少有的按拓展版本分批付费的游戏

率并没有那么高，在现阶段反不如直接提供部分精华让用户看到价值可能更靠谱（靠近核心的亮点，例如内容量足够让玩家了解游戏主题，并以现有内容去向玩家证明，购买版本的游戏可玩性会像体验版一样精彩，并且能够更有满足感），然后在此基础上重新提交付费下载版本。

包月付费模式

一般包月付费模式可分为两个层面：一个层面针对特定游戏下载批量包月，只要在下载包范围内就不再做限制，这个比较早之前就存在，诸如沃达丰（Vodafone）在欧洲推出的手机游戏自助服务，支持功能手机用户每月支付5欧元，下载体验多款基于Java ME技术的游戏；例如Big Fish Games去年传闻的开放包月订阅功能（《Big Fish》提供完整的游戏包，用户在付费后可在不同的游戏间进行切换，但最终貌似没有实现）。

另外一个层面是针对游戏内消费的包月订阅，这种现象可能更为明显，诸如Google Play向开发者提供的内置订阅服务（VIP Glub服务）或者更早之前的案例DeNA旗下的《Mobage-town》也曾推出虚拟消费按月结算方案（用户在该平台购买虚拟道具时可按月结账，例如最低6美元）。



Rovio依靠小鸟的周边产品取得了不错的收益

周边营收模式

对于规模化的手机游戏公司而言，已很难用单纯的游戏属性进行界定，相反可以看到类似Rovio之类的公司有相当一部分人员从事的工作是和游戏的研发或者发行没有关系的，他们所着重深度挖掘的是游戏在品牌化以后如何丰富好整个游戏的线上线下产品线，美国玩具生产商Commonwealth Toys & Novelty就曾做过预测2012年它将售出价值4亿美元的《Angry Birds》周边产品（按照ArctiStartUp统计，Rovio今年收益可能达到1.5亿欧元）。

同样在App下载上遥遥领先的Outfit7也在玩具、动画等周边上不懈发力，聘请到前迪士尼主管消费品的Andy Mooney执掌整体的周边产品策略；而近期在App Store和Google Play独领风骚的《Where is My Water》也有属于自己的雄心壮志，按照Tim Fitzrandolph的说法他们意在全新的领域和角色上去匹配迪士尼既有的荣誉将Swampy（小鳄鱼的名字）形象送入迪士尼乐园（目前也有周边玩具）；至于ZeptoLab日前与Ape Entertainment合作推出以《Cut The Rope》中的小怪物Om Nom为主题的连环画和漫画小说也相当讨喜。



郑金条

游戏邦负责人，游戏邦主要关注和解析国内外社交游戏和手机游戏领域，并定期做深度行业阐述。

责任编辑：陈博（chenbo@csdn.net）

揭开 ipa的面纱(上)

简述App逆向分析工具与过程

文 / 张超

iOS & Android DevCamp是CSDN独立品牌的移动开发技术会议，2012年7月27~28日首先在北京举办。在本次大会上，友盟SDK Team Leader张超分享了App逆向分析工具与过程。本期文章将主要介绍ipa的基本结构，ipa中常见的文件格式的使用，以及一些iOS上特殊的处理的恢复方式等内容。

在App Store上经常会出现一些令人惊艳的App来颠覆排行榜的排名，那些挑战人类想象力的交互创新，就在这小小的屏幕上不断诞生。不可否认，我们学习的过程就是模拟、掌握方法，再结合自己的思想，用我们的方式表达出来。本文就会带大家逐一击破其中的奥秘，探寻ipa的外表下，每个App的真正组成。

什么是ipa

通过Wikipedia的.ipa(file extension)词条我们可以知道，ipa是iPhone Application Archive，也就是iPhone应用程序包。我们对其进行一些检验操作即可知道ipa文件的真身。

$ file Twitter.ipa

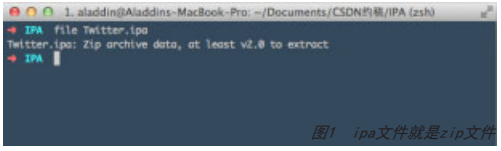


图1 ipa文件就是zip文件

由输出结果可知，其实ipa文件就是一般的zip文件，我们可以进行解压，得到其内部文件。

$ unzip -q Twitter.ipa -d Twitter

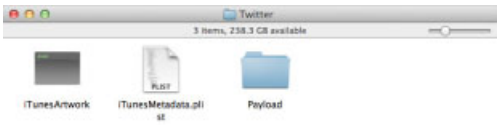


图2 ipa文件内有两个文件，一个文件夹

其内部结构比较简单，如图2所示可以看到ipa文件内拥有两个文件，一个文件夹。

- **iTunesArtwork**：512×512的PNG图片，用于在iTunes中显示。
- **iTunesMetadata.plist**：Apple PropertyList文件，基于XML格式的一种扩展格式，通常苹果将其用做描述性的文件。本文件用于记录购买该App的一些基本信息，及部分购买该App的用户信息。
- **Payload文件夹**：文件夹中存储了该App的软件包内容。

我们研究的重点是在Payload文件夹中存放的软件包内容。

App软件包分析

在Mac OS X上，对于所有后缀为.app的文件都会认作是Mac OS X上的软件，默认双击就是去尝试执行该软件。但实际上，我们拆出来的这个.app文件是不能够在Mac OS X上执行的，所以在系统的默认状态下显示的是一个不可执行的应用程序的图标。如果使用过一阵子Mac OS X的读者应该有经验，这个.app文件实际上是一个文件夹，我们也可以通过file命令去验证这点。

$ file Twitter.app

对于.app文件夹中的文件我们可以分为以下三类：资源型文件、数据型文件和可执行文件。

各种类型文件的研究方式不同，使用的工具和期待也不同。对于资源型的文件我们期待的是看到它本身的内容。无论是音频还是视频，还是图片，我们都需要一个直观的方法来观察到这些特殊数据所表达的内容。对于数据型的文件我们期待的是看到其数据结构，看到其内容，观察其他App对数据的管理方式，存储方式。而对于可执行文件，我们的期待则是能够看到一些细节的实现方式，对我们有更深层的指导意义。

资源型文件

资源型的文件包括xib文件、图片、音频等多媒体文件，这些文件基本上处于不会变更的状态，并不是通常意义上那么直接的数据，都具有自己特定的类型和格式。我们在这里着重讨论多媒体文件部分，因为这些是视觉直接相关的，大家通常对一个App感兴趣，大多数都是被它的交互或视觉所吸引，勾起了更多的欲望。

出现在.app文件夹中的多媒体文件不外乎视频、音频和图片。iOS支持的默认支持的音频格式有mp3、aiff、caf等，默认支持的视频格式有mp4、m4v、mov等容器装载的，编码为H.264的视频文件。一般的App比较少去对这两类文件其进行加密或进行一些预处理，除非是那些版权敏感的App。例如视频播放App和音频播放App，它们需要对缓存的格式做特别的预处理后再保存，从技术角度提高了视频版权的保护。甚至有些对版权控制很好的软件，会针对内容进行特定格式的编码和加密，做成自定义文件格式，以保证其版权的完整性。

针对没做处理的音频和视频，我们可以直接查看这些文件的参数（大小、编码方式、容器的选择）以及实际播放效果，以帮助我们更好地准备我们自己的内容。存在于软件包中的视频文件比较少见，因为它会增加App的大小，所以目前存在App中比较多的视频资源大部分是比较小的开场视频，或者介绍性的视频。这些视频的质量一般也不会很高，我没有见到谁针对视网膜和非视网膜准备两套视频文件，一般情况下，大家都会权衡一个视频质量来提供一份视频即可。

图片在iOS中算是比较特殊的一类多媒体文件。图片资源使用频率较高，而图片资源的大小会直接影响I/O的时间，因此iOS系统默认针对PNG这一推荐在GUI中使用的图片类型进行了优化和压缩，以保证渲染效果相同的情况下可以减小I/O和渲染时间。正是这个优化操作，导致了我们直接在ipa中的PNG图片无法直接在其他平台上正常显示。

优化PNG图片转化

优化PNG图片转化这个话题，曾一度成为业界热烈讨论的话题。在优化图片转换这方面的实现是玲琅满目，我会在这里介绍几种比较有代表性的实现。

基于优化算法做逆向还原

关于优化PNG的算法，在网上有详细的分析（http://iphonedevwiki.net/index.php/CgBI_file_format），这里不再进行赘述了。基于此方法的实现优势是跨平台，不受限于本地的操作系统环境，可以用相应的语言来实现，按需索取。缺点是如果苹果改变了一些PNG优化算法，这个就立刻失效了。

Python版本的实现：[ipin.py](http://www.axelbrz.com.ar/?mod=iphone-png-images-normalizer)

Ruby版本的实现：[iphone-png-normalizer](https://github.com/swcai/iphone-png-normalizer)

利用优化命令本身的逆向功能

在Xcode的编译后，复制资源时，会进行PNG优化的工作，如果不希望进行此操作，可以在Xcode中对相应的target进行设置。在Build Settings中将Compress PNG Files设置成NO即可。此处Xcode调用的优化命令其实是由Apple优化过pngcrush命令来完成，这个命令也有还原图片的参数：

```
$ pngcrush -revert-iphone-optimizations -q OptimizedPNG.png RevertedPNG.png
```

也有不少脚本是针对这个命令的调用进行封装的，这样的优点和缺点都比较明显，优点是无论苹果怎么修改算法，通过这个命令都可以还原回

来。缺点是只能通过安装了Xcode的Mac系统来完成这一操作，被局限在单独的平台，如果是作为一些线上服务的话，就比较难实现，Mac的服务器相对成本较高。

利用iOS系统

最后一种方式是通过iOS系统来实现这一过程的还原。这个方法相对技巧性比较大，发现也比较偶然，我在一次尝试中发现了这种方式。任何被优化过的PNG图片，只要被iOS系统读入之后，都可以正常显示的图片。系统可以正确识别其优化的结果。但从系统保存到沙箱中时，为了保证PNG图片可以正常与外界对接，保存出去的就是普通的PNG格式的图片。因此我们可以通过iOS模拟器来完成这个工作，保存到我们的Mac电脑中。使用这个方法的工具有：

- Objective-C实现：[UIKit-Artwork-Extractor](https://github.com/0xced/UIKit-Artwork-Extractor)
- Objective-C实现：[Normalize-iPhone-PNG](https://github.com/iAladdin/Normalize-iPhone-PNG)

这种方式的限制条件比较多，但稍加工也可以实现像UIKit-Artwork-Extractor一样，成为很方便的工具。

商业软件

除了这些开源实现之外，还有一些系统插件和商业应用是专门做这个的，Crunch是一款商业化比较好的App，专门实现提取ipa资源的功能。它提供了类似UIKit-Artwork-Extractor的功能，会去访问本地的iTunes媒体库中的iOS App，你可以直接选择任意App进行资源获取，操作便捷。

PVR/PVRTC/PVR.CCZ游戏资源图片转换

在图片资源中，除了PNG做了特别的处理之外，iOS系统本身还支持一些特殊的格式，例如PVR系列的格式。PVRTC格式全称是PowerVR Texture Compress的缩写，PVR则是对PVRTC的缩写。该图片格式是由Imagination Technologies公司对GPU运行进行优化后的高压缩比的图片格式。对于游戏开发而言，使用这个格式会更加高效，加上本身一些游戏框架对它的支持比较好，例如Cocos2D，它更倡导大家使用这个格式。Cocos2D社区更是针对这个格式进行了进一步的优化，使得可以在更小的尺寸下，保证比较好的图片质量，并提供了代码和工具层面的支持，使得PVR.CCZ格式也逐渐流行起来。

PVR系列格式在Mac上并不是原生支持的，因此这里也需要一些工具或插件来完成这个功能。

- 开源实现

quickPVR是一个开源的实现，实现了基本的PVR格式的解析过程，并把它写成一个QuickLook插件，这样使用起来也比较方便。建议程序员可以看看这个代码，并针对性地去写一些小的功能，来协助自己完成一些高级功能。例如最直接的功能：将它导出成PNG图片。

- 商业实现

这个商业实现也有一些，此处只介绍一个：TexturePacker，对这三种图片格式都支持得比较好，可以很方便地生成PVR、PVRTC、PVR、CCZ，界面简单，自带了一个viewer，可以很方便地查看PVR系列的图片。一般的游戏拼图的工作完成得很好，输出的文件可以直接在程序中方便地处理。该App作者的其他工具大家可以尝试使用下，作为游戏开发者应该会觉得有所帮助，对动画的工具支持，以及对物理引擎的工具支持，都用起来很方便。

数据型文件

对于App中的数据，可以按照数据所在的位置来分类进行描述，如图3所示。

- App Bundle内
- 沙箱内
- 沙箱外

我们刚刚所描述的资源型文件大部分处在App



图3　App数据中按所在位置分类

Bundle内，这些文件的特点是稳定，基本上不会被修改，也没有权限进行修改。App编译，打包之后的文件基本上就是这些文件。而沙箱内的数据，大部分是用户和App共同创造的数据，这部分数据是与App的实现和App的运行有很大关系的数据，App的数据持久的位置就是沙箱内。沙箱外的数据是通过App的运行时与App程序进行互动的数据，这里大部分是网络数据，我们今天着重讲解关于沙箱外数据，也就是网络数据的分析。

网络抓包与分析过程

在目前的iOS的客户端产品的网络交互的基本上都是使用常用的数据格式的HTTP协议的交互过程，当然目前的聊天客户端和涉及安全性比较高的客户端软件的交互也会像传统的桌面客户端一样，去自己做通信协议，或走HTTPS来增强数据的安全性。由此可见，大部分客户端软件的安全系数并不高。从做App的角度来看，这样做的好处便于维护和调试，开发过程和后续的维护成本都比较低廉。但从用户和系统的安全考虑的话，需要在安全性上做更多的改善。

针对目前的大部分的实现方法，也就是HTTP的网络交互为主的情况，我们的网络分析工具也会针对这个进行选择。目前Mac上常见的抓包工具有：

- Cocoa Packet Analyzer
- Charles
- Wireshark
- paros

其中只有Cocoa Packet Analyzer是原生的App，使用的是Objective-C实现的。其他的均不是原生的。均衡来讲，最强大的是Wireshark，作为可以分析各种协议的老牌抓包工具。从Linux/UNIX系迁移来的开发者应该对这个很熟系。缺点也很明显，易用度没有那么好。Cocoa Packet Analyzer仅限于能用，尽管是原生软件，但整个App却做的没有那么的用心。paros，是Java写的，免费的App，可以无限制地使用，但无论是功能上、交互上，都仅限于能用，并没有到推荐的程度。

我个人十分推荐Charles，软件十分易用，虽然是收费软件，但只是在使用时长上有限制，功能上是完整的。尽管是Java App，但其GUI的设计和交互的展现都很好。仅仅针对HTTP和HTTPS请求的抓取，但做得十分易用，在这点上算是秉承了Apple所倡导的方针，提供某一场景或需求的解决方案，十分专注。Charles的工作原理是自己做一个代理，将系统的网络代理设置为它的起始端口，那么从它经过转发的网络请求自然都可以被它分析到。

使用Charles进行HTTP请求的分析

使用Charles进行普通的HTTP请求的抓取不需要什么设置，在首次启动时，Charles会提示你为它进行授权，允许它修改系统的代理设置。在授权之后，在普通的使用上，就不需要设置了。如果是抓Mac的网络请求，只需要确定Recording是启动的状态，且代理设置是启用的就可以了。

针对iOS设备的抓包也可以通过它来进行。模拟器的请求默认走的是Mac的网络代理设置，所以如果你要抓的是模拟器的包进行调试的话，只要确认Charles是工作状态的，就可以了。如果是针对iOS物理设备的抓包，则需要更多的设置。

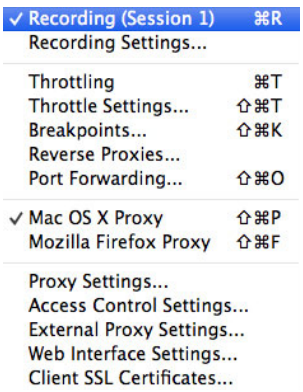


图4　确定Recording是启动状态

首先在Mac上启动Charles，确保Charles工作正常。在Mac上，打开系统偏好设置→共享→启动网络共享，这时候需要将Wi-Fi作为分享出口，Mac的上网方式（一般使用以太网接口，也就是将有线网络分享为无线网络）的接口作为分享源。设置完成后，Wi-Fi的图标会在系统状态栏中会变成分享标志。

在iOS上，打开设置，选择Wi-Fi连接，选择你的Mac的名称，如设置了密码，则请在连接时输入正确的密码，以保证Wi-Fi连接成功。若出现连接不成功的现象，可以去Mac上重设一下共享Wi-Fi的密码，然后再开启Wi-Fi，一般都可以解决。关键的一点，在iOS上确认，该Wi-Fi的代理设置是此热点的路由器地址，端口为8888（此为Charles默认端口，如有修改，请对照Mac上的设置进行设置）。



图5　Charlse的抓包结果的展示

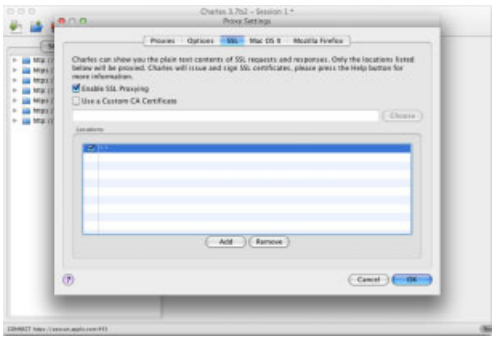


图6　开启SSL的监听

使用Charles对HTTPS请求分析

使用Charles进行HTTPS的请求抓包，需要下载由Charles官方提供的CA证书。导入到本地Keychain中的你的Login keychain中。接下来，就需要在

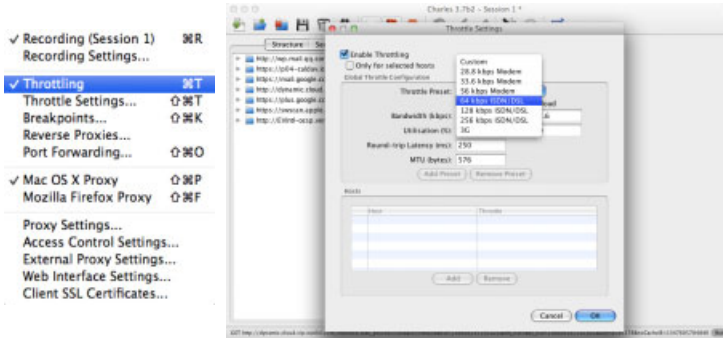


图7　设置不同的参数来模拟各种网络环境

Charles的Proxy设置中，开启SSL的监听，输入指定的URL可以为特定的URL进行监听。其URL规则也支持模糊匹配。

使用Charles进行特殊网络的模拟

除去在研究别人的App时可以Charles用进行抓包，在模拟一些特殊网络时，也可以用Charles进行模拟。在Proxy菜单下可以使用Throttling选项，通过设置不同的参数可以来模拟各种网络环境，也可以方便地使用它的一些预设模拟。这个对于我们调试自己的程序在极端网络环境下的运行状况还是很方便的。

总结

本期我主要介绍了ipa的基本结构，ipa中常见的文件格式的使用，以及一些iOS上特殊的处理的恢复方式。数据方式方面则介绍了如何在对iOS App进行抓包分析以及调试的方法。下期文章将会对如何从ipa中的二进制文件获取更多的信息做详细的介绍。



张超

资深iOS 专家，iOS创业者，目前在友盟担任SDK Team Leader，是创新工场和友盟的早期团队成员。

责任编辑：陈博（chenbo@csdn.net）

Android软件安全开发实践

文 / 肖梓航

上一期中，我们讨论了数据存储、网络通信、密码和认证策略等安全问题和解决方案，本期将继续从组件间通信、数据验证和保全保护等方面来实践Android软件安全开发之路。

组件间通信

组件间通信的安全问题是Android所独有的，也是目前软件中最常出现的一种问题。

我们先回顾一下组件间通信机制。Android有四类组件：activity、service、broadcast receiver和content provider。在同一个软件之中或不同软件之间，前三种组件使用Intent相互调用，使用ContentResolver对象访问content provider，共同实现软件的功能。使用Intent，可以显式或隐式地调用：

- 显式（explicit）：调用者知道要调用谁，通过组件名指定具体的被调用者；
- 隐式（implicit）：调用者不知道要调用谁，只知道执行的动作，由系统选择组件处理这个请求。

如下面的代码所示：

```
Intent intent = new Intent();

// 下面是三种显式调用
intent.setClass(getApplicationContext(),
  theNewActivity.class);
intent.setClassName("com.secmobi.App",
  "com.secmobi.App.theNewActivity");
intent.setComponent(new Component("com.secmobi.App"
  , ".theNewActivity"));

// 下面是一种隐式调用
intent.setAction(Intent.ACTION_VIEW);

startActivity(intent);
```

无论是显式还是隐式，如果要跨应用调用，还需要被调用的组件是对外暴露的。默认情况下，service、broadcast receiver和content provider是暴露的，申明了Intent-filter的actvity也是暴露的。

抽象地说，组件A要调用组件B，以期待B完成某个功能；它可以发送一些数据给组件B，也可以获得B执行后的返回结果。在这个模型中，问题出现在A和B之间不一定互相可信。

如果B是暴露的，任何软件都可以调用它，包括攻击者编写的软件。攻击者可能但并非总能成功：

- 直接调用暴露的B，以获得其执行结果；
- 构造特定的数据，并用于调用暴露的B，从而试图影响B的执行；
- 调用暴露的B，并获取它执行完返回的结果。

如果A用的是隐式调用，任何软件都可以实现它的action从而响应调用。攻击者可能（但并非总能成功）：

- 构造伪造的组件C，响应A的Intent，以读取A要发给B的数据；
- 构造伪造的组件C，响应A的Intent，弹出虚假的用户界面以展开进一步攻击（例如钓鱼）；
- 构造伪造的组件C，响应A的Intent，返回伪造的执行结果。

这样说可能比较抽象。下面我们对这两种情况分别讨论。

组件暴露的问题

看一个例子。在一个第三方深度定制的ROM中，预装了名为Cit.apk的软件，用于手机的硬件测

试。它的AndroidManifest.xml局部如下：

```
<receiver android:name=".CitBroadcastReceiver">
  <intent-filter>
    <action android:name=
        "android.provider.Telephony.SECRET_CODE" />
    <data android:scheme="android_secret_code"
        android:host="64663" />
    <data android:scheme="android_secret_code"
        android:host="284" />
    <data android:scheme="android_secret_code"
        android:host="6564" />
  </intent-filter>
</receiver>
```

可以看到，它申明一个名为.CitBroadcastReceiver的receiver，响应名为android.provider.Telephony.SECRET_CODE的action，并且指定了URI格式。

再来看这个receiver的代码片段（下面的代码是我反编译得到的，不一定与软件源码完全一致）：

```
if("284".equals(paramIntent.getData().getHost())) {
  m_logFileName =
      CitUtils.getLogFilePath("bugreport");
  // ......
  asyncExecute(new Runnable() {
    public void run() {
      try {
        String str = CitBroadcastReceiver.TAG;
        String[] cmd = new String[3];
        cmd[0] = "bugreport";
        cmd[1] = ">";
        cmd[2] = m_logFileName;
        CitUtils.rootExecProgram(str, cmd, true);
      }
      // ......
    }
  });
}
```

可以看到，当调用这个receiver，并且提供的URI中host字段为284时，会以root权限调用本地的bugreport工具，并将结果输出至m_logFileName指定的文件中。

默认情况下receiver是暴露的，因此这个receiver可以被其他软件调用，代码如下：

```
Intent intent = new Intent();
intent.setAction("android.provider.Telephony.SECRET
    _CODE");
intent.setData(Uri.parse("android_secret_code://
    284"));
sendBroadcast(intent);
```

当这四行代码执行时，就会触发CitBroadcastReceiver的那段代码。从上下文看，输出文件m_logFileName位于SD卡，任何软件都可以随意读写。因此，攻击者可以获得bugreport的输出结果，其中包含大量系统数据和用户数据。

请注意，在这个例子中，攻击者的软件不需要任何特殊权限，尤其是不需要root权限。这种由于组件暴露获得额外权限的攻击，被称之为permission re-delegation（权限重委派）。

怎么避免由于组件暴露产生的安全问题？有的组件必须暴露，例如入口activity，或者确实对外提供服务或跨软件协作；但也有的组件没必要暴露。接下来我们分别讨论。

不需要暴露的组件

再次回顾，默认情况下，service、broadcast receiver和content provider是暴露的，申明了Intent-filter的actvity也是暴露的。如果它们只被同一个软件中的代码调用，应该设置为不暴露。很容易做到——在AndroidManifest.xml中为这个组件加上属性android:exported="false"即可。

需要暴露的组件

如果组件需要对外暴露，应该通过自定义权限限制对它的调用。

首先，在实现了被调用组件的软件的AndroidManifest.xml中自定义一个权限：

```
<permission android:name=
    "myApp.permission.CALL_MYSERVICE"
  android:label="@string/permlab_myService"
  android:description="@string/permdesc_myService"
  android:permissionGroup="android.permission-
      group.SYSTEM_TOOLS"
  android:protectionLevel="signature" />
```

接下来，为被调用组件添加这个权限限制，即在AndroidManifest.xml中为这个组件添加android:permission属性：

```
<service android:name=".MyService" android:
    permission="myApp.permission.CALL_MYSERVICE" />
```

另一种方法是在组件的实现代码中使用Context.checkCallingPermission()检查调用者是否拥有这个权限。

最后，要调用这个暴露的组件，调用者所在的软件应该申明使用这个权限，即在AndroidManifest.xml中添加相应的use-permission申明。

进一步地，还可以将这种组件暴露的需求分为两种情况。

- 如果这个组件只打算给自己开发的其他软件使用，而不希望暴露给第三方软件，在定义权限时，protectionLevel字段应该选择signature。这种设置要求权限使用者（即调用者）与权限定义者（即

被调用者）必须由相同的证书进行签名，因此第三方无法使用该权限，也就无法调用该组件。

■如果这个组件要暴露给第三方，则protectionLevel应使用normal或dangerous。此时，任何软件都可以使用该权限，只在安装时会通知用户。考虑到用户一般会忽略权限提示，此时自定义权限实际失去了保护效果，我们依然要仔细审查该组件的代码，避免出现能力泄露或权限重委派等问题。

此外，对content provider，可以更细粒度地为读取数据和写入数据设置不同的权限，对应的manifest标签分别为android:readPermission和android:writePermission。

隐式调用的问题

隐式调用的主要问题是被劫持，或Intent携带的数据被读取。

为了劫持调用，攻击者可以实现一个恶意的组件，申明相同的Intent-filter。在多个组件都可以响应同一个Intent的情况下，如果是调用activity，系统会弹出界面要求用户对多个软件做出选择，攻击者可以模仿真实软件的图标和名称吸引用户点击；如果是调用service，系统会随机选择一个service；如果是调用receiver，系统会逐一地将Intent发给这些receiver。

劫持了调用后，攻击者可以（但并非总能成功）：

- 启动一个虚假的软件界面，展开钓鱼攻击（例如要求用户输入账户密码）
- 读取Intent携带的数据
- 拦截broadcast的进一步发送，导致真正的receiver无法收到消息，从而拒绝服务
- 如果是用startActivityForResult()调用了虚假的activity，可以返回恶意或虚假的结果给调用者
- 执行其他恶意代码

限于篇幅，对这些情形我们不提供示例代码。来看一下怎么解决这类问题。

不需要隐式调用

除了基于Intent类中已有ACTIONs的隐式调用，绝大部分隐式调用都属于这两种情况：同一软件中不同组件的调用；同一开发者不同软件间的调用。

这两种情况下，事实上，开发时都已可以确定要调用的组件是哪个。因此可以避免隐式调用，改为基于组件名的显式调用。

需要隐式调用

发送broadcast除了使用sendBroadcast(Intent)，还有一个方法是sendBroadcast(Intent, String)，它的第二个参数可以指定接受者需要的权限。

如果是调用activity或者service，目前我所知，还没有简单的方法实现接收者的权限限制。在Android文档中提出可以自定义Binder和AIDL实现通信双方的互相验证，但真正实现并不容易，也不为官方所推荐。

数据验证

无论是客户端还是服务器，在处理外部获得的数据之前，都应先判断和验证数据的有效性。这里主要指是否包含畸形的数据。

在Web开发中，服务器需要对用户提交的数据进行有效性验证，否则很容易出现众所周知的SQL注入等攻击。在移动开发中也不例外。虽然客户端与服务器在底层通信协议上对用户是透明、不可见的，但开发者不应因此就假设双方传输的数据永远会和预先设计的一致。类似的，在读取用户从UI元素输入的输入、读取存储在本地的数据后，使用前也应进行有效性验证。

软件版权保护

攻击者对Android软件进行逆向分析，除了寻找其中的安全漏洞，还可以直接攻击软件本身。

- 破解软件的收费机制、License验证或功能限制；
- 修改软件代码，去掉广告库，或者修改广告库（一般是改变推介ID字段），或者增加广告库，然后重新打包并分发；
- 重新打包，植入恶意代码并分发；
- 逆向分析，学习软件特色功能的实现方法，或者获得可复用的代码；

可以采取多种措施来增加破解、修改和逆向分析的难度，减少被攻击的可能。这些措施包括：

- 使用代码混淆工具，例如SDK带的ProGuard以及其他Java混淆器。
- 采用NDK开发核心模块。
- 使用官方或第三方的软件保护方案，例如SDK的android.drm包、Google Play的软件许可（Application Licensing）支持。
- 去掉开发时的调试代码，关闭调试开关，删除多余的Log代码。

然而，采取了这些措施并不等于就万无一失了。例如，用NDK开发的Native代码，也可以使用IDA Pro及其插件来反汇编、反编译和调试；用Google的DRM方案，也有AntiLVL这样的破解工具。理论上，现有防御手段都可能找到方法继续攻击，其价值只是提高攻击难度和成本。

总结

已出现和将要出现的威胁

到目前为止，在学术研究以外，针对Android软件漏洞的攻击只出现一起——劫持国外多个社交网站客户端登陆会话的黑客工具FaceNiff。但随着攻击者制造传播恶意代码的成本增加和收益降低，以及移动终端隐私数据逐渐成为地下产业链的交易资源，针对Android流行软件漏洞的攻击在未来几年之内几乎一定会出现并爆发。

统一的安全模型

限于篇幅，本文只介绍了几种常见且简单的安全问题，还存在许多我们知道的、还不知道的漏洞。如何找到和解决这些问题？

回顾已介绍的内容，我们可以发现它们有类似的安全模型：通信双方的信任问题。

- 在数据存储中，读写数据的代码和存储在本地的数据互相不可信。
- 在数据通信中，发送者和接受者互相不可信。
- 在登录认证中，发起认证请求的用户的和接受认证请求的服务器互相不可信。
- 在组件间通信中，发起Intent的组件与接收Intent的组件互相不可信。
- 在数据验证中，处理数据的模块不能相信产生数据的源。

面对将来的问题，我们也可以尝试抽象出这种模型，区分互相不可信的实体，然后在不可信、不安全的基础上，尽可能地实现相对的可信和安全。

进一步学习和行动

Android的开发文档Best Practices: Designing for Security和源码文档Tech Info: Security分别从开发和系统实现的角度介绍了系统的安全机制。另外，viaForensics提供了名为42+ Best Practices: Secure mobile development for iOS and Android的在线教程，更详细地介绍了移动软件面临的安全威胁，并给出了安全开发实践策略。

社区方面，从Android安全开发的角度，StackOverflow并不一定是很好的选择——其中一些最佳回答没有考虑安全，直接使用可能产生问题。Google Group的anroid-security-discuss讨论组则更为专业和准确。OWASP成立了一个Mobile Security工作组，目前已发布Top Ten Mobile Risks等多份白皮书，并举办了AppSec会议。这个工作组的效率虽然不高，但产出质量非常棒。

学术方面，2011和2012年的四大会议及其workshop上均有移动软件漏洞挖掘和攻击阻止的论文出现，从它们的related works部分可以综合快速地了解学术界的思路。

目前的移动开发还没有形成如此成熟的体系，这也许与其轻快敏捷的互联网产品开发风格有关。但我相信，真正实效的移动软件安全开发，最终依然会融合到需求分析、系统设计、开发实现、测试验证、部署运维等每一个环节，从而与PC平台的SDL殊途同归。



肖梓航

安天实验室高级研究员，主要方向是移动反病毒和移动软件安全，发起或参与了多个移动安全开源项目。创办了网站secmobi.com
博客：http://claudxiao.net。

责任编辑：陈博（chenbo@csdn.net）

Android敏捷开发指南

文 / 王哲

本文延续上期话题，深入到测试、持续集成和部署等环节，紧密结合移动开发方法和技术，围绕Android平台的开发讨论提供更高质量移动产品的解决方案。

通过清晰的架构实现测试驱动

通过《程序员》杂志9月刊文章的分析，我们可以看到，每一种工具都很难从完全意义上解决工程当中追求快速和高质量的要求。那么就需要通过整体架构实践，更好地解决这方面的问题。以下两种结构方法可供参考。

平台和领域的分层结构

如图1所示，这是一种最基础的分层结构。它将和Android平台相关的内容都包裹在了平台层，而将领域、状态、逻辑等和平台无关的内容完全隔离开来。创建项目时，将平台层和测试层分别对领域层产生依赖，以这样的方式达成对项目和对测试的独立构建。

这样做的好处是，领域层可以通过Java的JUnit进行完全测试，而项目各个类之间的因为有明显的项目级别分层，使得代码的考量更加清晰，方便于检验最核心关键的业务逻辑。且测试速度非常快，便于快速迭代。

但其带来的不便之处就是，随着项目代码的日益复杂，其与Android内部结合就越紧密，这时往往越难将业务逻辑很清晰的剥离。部分的业务逻辑将被分片散落在一些平台相关的调用当中。因此我们认为这样的一种层次逻辑更多地适用于简单的项目之中。

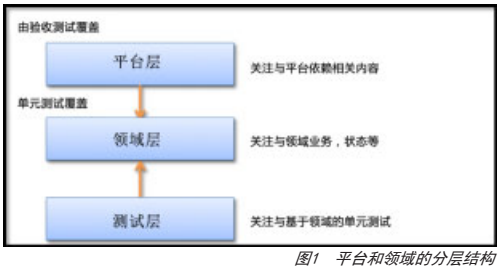


图1　平台和领域的分层结构

MVP（Passive View）架构

许多文章对Passive View是属于MVP还是MVC有各种不同角度的争论。这里我并不想过多地在词语上做过多纠缠，主要是希望通过描述这样的一种结构来树立比较好的对于Android开发的架构模型。

具体到Passive View的实现，如图2所示，描述了一个基本的原理。

随着项目的不断扩大复杂化，我们需要更加清晰化的代码架构。所以，Passive View对平台和领域的分层结构的层次结构有了更深层次的改进。引



图2 Passive View架构实现原理图

入Passive View模式的意义实际上还是从测试出发。出于对JUnit快速性能的青睐，所以依然尽量把层次划分为依赖Android内核的部分和非依赖部分。对于前半部分，可以用基于Roboletric的上层测试完成，相对应的后半部分则可以使用标准Java进行单元测试。这样做既加快了测试速度，又得以保障测试的覆盖率。

更多详细的架构资料，请大家参考《GUI Architecture》一文，相信会得到许多很好的想法。

测试驱动开发的实现

之所以要强调代码框架结构，最重要的原因就是寻找快捷方便的途径进行TDD开发。在上一篇文章中已提到了Android常见的集中单元测试工具。之所以将框架设计方式引入测试驱动开发过程，目的是将应用代码分开处理、分而治之。最大限度让各工具间扬长避短。最终实现高效率测试驱动的目的。

事实上，在目前的开发领域，我们也看到了许多正在应运而生的开发框架，例如Android Spring框架。这些都是下一步需要努力和完善的目标。

业务行为驱动的功能测试方案

如果说单元测试驱动开发帮助我们从开发角度完成了对代码质量的控制，那么基于业务行为的功能测试则为从业务到实现的统一、软件质量的保障提供了重要的解决方法。

功能（验收）测试：就功能测试而言，所指基本上就是测试科目中常提到的黑盒测试。只不过这里加上了更多具有上下文的信息，使得一次完整的黑盒测试更具有实质上的功能意义。

在Android的开发中，通常是模拟真机或模拟器的用户操作（例如点击或滑动），来检测操作的结果是否符合预期。这个过程可以很好地代替人工的操作，更快速和大量完成重复性的测试工作，特别是在发展较快、平台和系统版本覆盖较广的项目中，可以起到节约人力成本、提高测试覆盖与准确性的作用。

在常用的工具中，可以选取上一篇文章中介绍过的Robotium或Native Driver来实现。

前者社区发展速度快，实现功能相对更加完善，例如实现了一些手势操作效果等。不过问题是Robotium的组织目前正在倾向闭源化，有些最好的新功能已无法在开源版本中寻找到。

后者的原理是通过植入应用后门的方式，实现对应用的发出指令和进行资源获取。其物理模型比较适合不能再目标设备安装测试应用，或者希望远程控制的情况。开源状况良好，只是近一年，Native Driver的团队把重点发在其他平台上，对Android的更新比较慢。

基于场景的的跨平台验收测试：基于场景的验收测试，好处是可以在业务人员和开发人员之间建立验收机制。通过一系列关于场景的自然语言描述，完成测试的原始编码。Cucumber是一个在行为驱动开发（BDD）领域比较常见的工具。其基本方法是通过执行文字形功能描述语言来实现对软件的自动化测试。图3展示了一个常见的Cucumber测试场景。在Android开发中，通常是解析Cucumber，根据内容分情况拆分正具体的操作步骤。将这些步骤用基于NativeDriver的Java代码进行实现，通过NativeDriver从指令端到设备端的Android自动测试框架完成控制与校验的自动化测试。

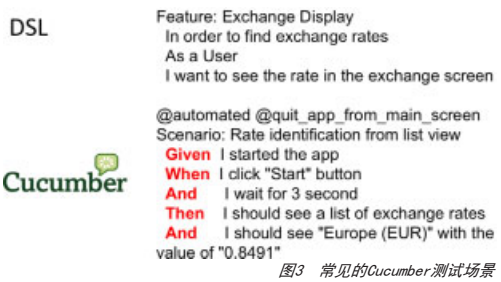


图3 常见的Cucumber测试场景

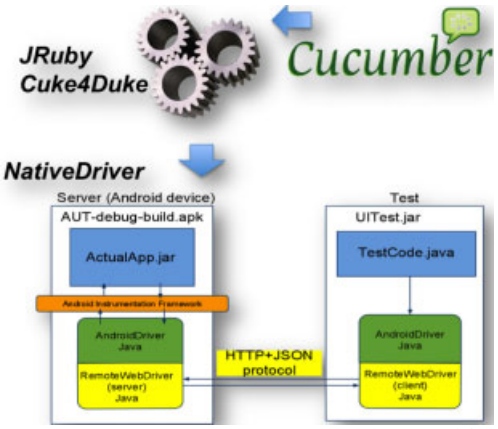


图4 Cucumber测试实例

当然，这些步骤也可以通过Robotium的远程控制（RC）调用来实现。其原理类似，如图4所示。

该方案还有额外的好处——用同样的方法，通过Cucumber可以实现对UIAutomation等其他平台

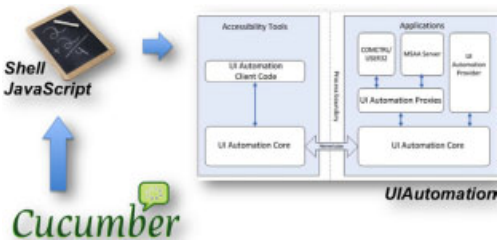


图5 iOS平台上基于Cucumber的跨平台测试实例

自动化测试工具的控制。实现一份用例，多个平台的测试解决方案。图5给出了一个iOS平台上的基于Cucumber的跨平台测试实例。

从持续集成到部署

Jenkins最基本的持续集成

在做好了各种测试实践后，需要考虑如何保证这种实践在每一次提交都产生效果，并且提供对代码库的不断保护。这里引入常用敏捷实践中持续集成的概念。

持续集成主要是为了能够更好为产品在整个开发过程中提供交付保障。相对比传统服务器领域，移动开发中，碰到最多的问题是环境的不同与测试包的相对封闭。以前通过改一个配置实现服务



图6 利用相关应用包完成对不同环境的开发与测试

器实时测试修改的方法，很难直接应用于移动开发。此外，作为整个服务边缘的移动应用，往往会遇到大量不同的测试环境。

所以如图6所描述的，需要在持续集成服务器上部署相关的应用包，完成对不同环境的开发与测试。这样的测试通常是由持续集成服务器引导的不同物理移动设备来完成，以保证最终测试包的可交付性。尤其适用于对于网络和性能等相关的测试。

当然，如果细心的话，也可以看出图6中对于产品版本与配置管理的一些端倪。

OTA自动化部署

持续集成帮助我们有了稳定的应用包。有了这些包，我们现在就差最后的一步——自动化部署。在移动的真实交付中，由于安装包需要下载到移动终端完成。所以自动化部署也必须要模拟这一过程完成交付。

这里我们推荐使用持续集成服务器关于邮件服务推送的应用插件来完成。具体的过程通常为，在完成编译并顺利通过测试后，持续集成服务器会根据制定终端列表，推送测试内容以供安装。推送者收到请求后，可以根据需要自行安装相应apk文件，达到部署效果。

这样做打通了移动开发最后一步的障碍。并有效沟通了管理、设计、业务与测试的各个部门人员。持续集成管理员还可以通过管理安装邮件列表的形式，自行选择不同版本，环境安装包对不同相关人员的推送。

让调试可持续化

有了开发，有了一系列持续化集成作业，有了作业完成的自动化打包部署，貌似可以休息了。但正如DevOps概念所说，虽然做了许多努力，但我们依然承认会有错误的出现，那么如何尽早发现错误并准确定位，最终完成就显得尤为重要。在不同阶段，我们尝试使用不同的方法来进行调试。下面介绍我们的两个实践方法。

- **开发后调试**。旧有的logcat是一个很好的调试工具，它在开发中起到了重要的作用，但对于做探索测试的QA们而言，大量繁琐的Debug信息往往令人看得眼花缭乱。

这时我们就需要一个更好的工具进行错误栈的准确定位，因此我们引入了Acra。这是一个开源工具包，其功能是自动化地将Android应用的错误报告发送到GoogleDoc或着Email等的可记录文本中。它的目的是为了帮助Android应用开发者在错误发生时收集行为和技术数据。

- **上线后的持续关注**当然在上线后，Google Play也为我们提供了一个非常有好的平台用以收集各种来自用户的反馈信息。其中包括下载量、评价、应用异常报告等。

在应用异常报告中，详细记录了整个错误堆栈的信息，更包括错误信息的总数量、周发生次数，以及版本时间等信息。因为Android平台设备性能、具体型号等多方面内容，并不一定所有的错误信息都一定去进行处理，但这个平台提供了一个持续关注上线情况的基础，为产品的紧急处理、故障修复和在开发提供了重要的信息来源。

这里还要强调，测试优先的TDD实践，在每一个错误修复前，都应该把测试补好，这是同样错误不会再次出现的最重要保障。



图7 移动循环开发流程

诸如配置管理、发布管理等也是必不可少的。我们总结了如图7所示的一个移动开发循环。通过对不同方面的配置技术实现，完成移动的全流程开发，供读者做更深的思考和方法演进。

总结

本文提出了移动开发过程中常见的问题，并分析了基于开发与维护过程中的一些具体实践方法，尝试从移动应用的开发技术角度提供解决方案。

除此之外，事实上和常见的软件开发一样，还有



王哲

ThoughtWorks咨询师。曾多次带领团队从事完成Android、iOS、Blackberry、Mobile Web、数据整合API等项目。致力于将敏捷先进的技术管理经验与移动产品紧密结合。

责任编辑：陈博（chenbo@csdn.net）

MySQL云数据库服务的架构探索

文 / 曹伟

虽然近两年来NoSQL发展迅猛，但MySQL凭借其成熟的中间件和运维工具及良性的生态圈，在互联网企业中的应用仍然非常广泛，并占有主导地位。本文将深入探讨依托MySQL平台设计和实现的UMP系统的架构探索经验。

MySQL作为一种低成本、高性能、可靠性良好而且开源的数据库产品，在互联网企业中应用非常广泛。例如，淘宝网就有数千台MySQL服务器。虽然近两年来NoSQL的发展很快，新产品层出不穷，但在业务中应用NoSQL对开发者来说要求比较高，而MySQL拥有成熟的中间件、运维工具，已经形成一个良性的生态圈。因此，在现阶段的应用中仍然以MySQL为主，NoSQL为辅。

在过去一年里，我们在MySQL托管平台方向做了大量工作，设计和实现了一套UMP（Unifield MySQL Platform）系统，提供低成本和高性能的MySQL云数据库服务。开发者从平台上申请MySQL实例资源，通过平台提供的单一入口来访问数据。UMP系统内部维护和管理资源池，以透明的形式提供主从热备、数据备份、迁移、容灾、读写分离和分库分表等一系列服务。平台通过在一台物理机上运行多个MySQL实例的方式来降低成本，并且实现了资源隔离，按需分配和限制CPU、内存和I/O资源，同时在不影响提供数据服务的前提下，支持根据用户业务的发展来动态扩容和缩容。

架构的演变

UMP系统第一版基于MySQL Proxy 0.8版修复了若干Bug，并对Proxy插件中管理用户连接和数据库连接的状态机流程进行了修改；编写了Lua脚本实现到中心数据库获取用户认证信息和后台数据库地址，来对用户进行验证；建立了到后台数据库的连接和转发数据包等逻辑（如图1所示）。

在开发和部署第一版的过程中，我们逐渐认识到几个问题。

首先，MySQL Proxy 0.8版对多线程的支持比较简单粗暴，多个工作线程共享同一个消息队列，同时监听着同一个socketpair通道。当有新事件进入消息队列后，socketpair会被写入一个字节，所有休眠中的线程都会被唤醒，去竞争一个互斥锁从消息队列中取任务。这种实现有几个问题：一是造成“惊群”现象，多个线程被唤醒但只有一个线程需要去完成任务；二是任务的CPU亲缘性比较差，在同一个状态机上触发的事件会在多个处理器上来回切换执行。此外，MySQL Proxy中还使用了全局Lua锁，同时仅允许一个工作线程执行Lua脚本（计划在0.9版本中改进）。因此，在多线程模式下，MySQL Proxy的性能远不能同CPU核数保持线性增长，甚至在16核上的性能还不如4核。而使用单进程模式时，一台物理机上需要部署多个进程才能有效利用机器的处理能力，但给部署、监控和服务的升级带来麻烦。

其次，由于MySQL Proxy的框架在功能上不容易扩展，所以实现用户的连接数限制、QPS限制及主从切换、读写分离、分库分表等功能比较困难。

最后，MySQL Proxy的社区近些年并不活跃，且C语言对开发者功底的要求比较高，很难要求团队所有成员协同开发出兼顾优雅和正确性的代码。

因此，我们决定用Erlang语言重新编写Proxy服务器，替换了原有的MySQL Proxy模块。目前，整个项目拥有5万行Erlang源码，3万行C/C++源码，2万行其他语言源码。

为什么选择Erlang语言

Erlang是一个结构化的、动态的、函数式的编程语言。常见的一种说法是Erlang是面向并发的（Concurrent-Oriented），这主要指Erlang在语言中定义了Erlang进程的概念和行为（本文中提到的“Erlang进程”都是指Erlang语言中定义的进程，以区分于大家熟悉的操作系统进程）。与操作系统的进程/线程相比，Erlang进程同样是并发执行的单位，但特别轻量级，它是在Erlang虚拟机内管理和调度的“绿进程”，即用户态进程（如图2所示）。举个例子，在关闭了HiPE和SMP支持的Erlang虚拟机中，一个新创建的进程占用的内存仅为309个字（Word，64位服务器上为8个字节）。其中233个字为堆空间（包含栈），创建和结束一个进程约耗时1~3微秒，而一个Erlang虚拟机中可以同时支持几十万甚至更多个进程。

说到Erlang语言，就必须提及OTP（Open Telecom Platform，开放电信平台）。OTP是用于开发分布式的、高容错性的Erlang应用程序的框架与平台。例如，一个Erlang节点连接并注册到Erlang集群上，发现集群中的其他节点，并与它们进行RPC通信，这些都在OTP里的Kernel服务中实现。OTP和Erlang语言关系如此紧密，以至于两者通常合称为Erlang/OTP，因此从严格的意义上来讲，应该说我们选择了Erlang/OTP来构造UMP系统。Erlang/OTP很好地抽象了开发一个分布式的、高容错性的应用程序所需的要素，包括网络编程框架、序列化和反序列化、容错、热部署。

为了支持并发，服务器端多采用多进程/多线程模型，即每个进程/线程处理一个客户端连接。但受限于操作系统资源，每台服务器可以处理的并发连接数并不高，且由于进程/线程上下文切换开销，系统性能会受到影响。而开发高并发、高性能服务器一般采用事件驱动的状态机模型，底层采用非阻塞I/O（Linux中的epoll，BSD系统中的

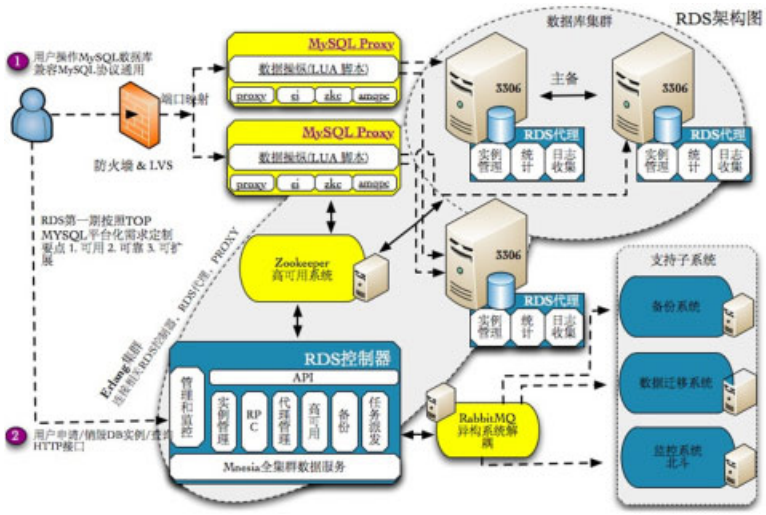


图1 UML系统的第一版（当时称作RDS系统）采用MySQL Proxy

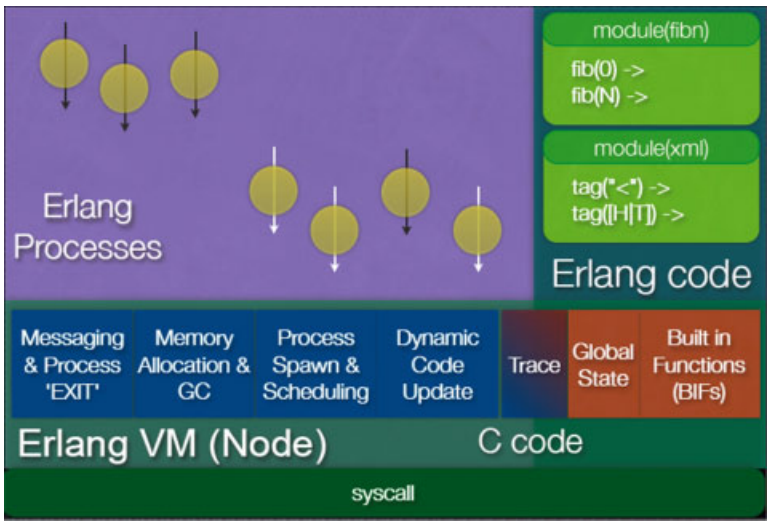


图2 Erlang的轻量级进程

kqueue，Java中的nio）或者异步I/O，或者采用异步的事件通知的I/O框架，例如C/C++下的ACE、boost::asio、libevent，Java下的MINA等。在业务层则使用状态机来表示每个客户端连接，通过I/O事件、超时事件驱动状态机进行跳转，每个进程/线程可处理成千上万个客户端连接。与多进程/多线程模型相比，虽然事件驱动的状态机模型并发量更大、性能更好，但把业务逻辑表达成状态机是一件困难的事情。相比之下，多进程/多线程模型中的业务逻辑可以实现为顺序执行的代码，开发起来要简单得多。

Erlang/OTP中的网络编程模型则结合了两者的优点，每个Erlang进程处理一个客户端连接，业务逻辑是顺序执行的。Erlang进程是极轻量级的，

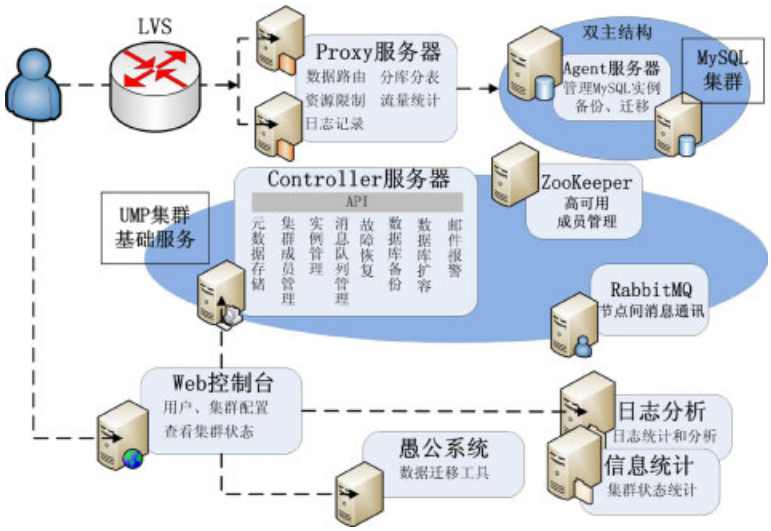


图3 当前UMP系统架构图

可以认为每个Erlang进程是一个状态机，堆和栈上的数据是这个状态机的状态。Erlang进程收到数据包或者其他进程发来的消息后执行处理例程，相当于状态机的跳转，因此也具有高并发和高性能的优势。

Erlang/OTP定义了“External Term Format”协议将Erlang数据结构与二进制字符串相互转化，并用C实现在Erlang虚拟机中，在进行跨节点通信时遵从这个协议。因此，开发者无须额外考虑序列化和反序列化问题。

在容错方面，Erlang进程的数据空间是相互隔离的，没有共享内存，因此一个Erlang进程崩溃不会影响其他Erlang进程运行，更不会造成Erlang虚拟机崩溃。OTP提供了监督树机制和heart模块，前者在监控到Erlang进程崩溃时进行故障恢复，后者在发现Erlang虚拟机失去响应时重启程序。

Erlang/OTP提供热部署方式，可以避免服务升级时造成不可用时间。此外，OTP还提供了一些在系统运行时观察系统状态的工具。例如lcnt工具，可以统计虚拟机内部的锁使用次数和冲突次数，指导系统的优化。

当前系统架构

在设计UMP系统时，我们遵循了以下几条原则。

- 系统对外保持单一入口，对内维护单一资源池。
- 保证服务的高可用性，消除单点故障。
- 保证系统是弹性可伸缩的，可以动态地增加、删减计算与存储节点。
- 保证分配给用户的资源也是弹性可伸缩的，资源之间相互隔离。

UMP系统中的角色包括：Controller服务器、Proxy服务器、Agent服务器、API/Web服务器、日志分析服务器和信息统计服务器。图3是当前UMP系统的架构图。UMP系统依赖Mnesia、LVS、RabbitMQ、ZooKeeper等开源组件。

Mnesia是OTP提供的分布式数据库，与MySQL NDB出自同门，都是20世纪90年代中期Ericsson为电信业务研发的数据产品。Mnesia支持事务、支持透明的数据分片，利用两阶段锁实现分布式事务，可以线性扩展到至少50个节点。

从CAP理论的角度来说，Mnesia更倾向于牺牲可用性来换取强一致性，属于CP阵营。但它也提供了脏读、脏写操作，可以绕过事务管理去操作数据，这时不保证一致性，有点类似于AP的系统。在工程实践中，我们用事务去修改关键数据（例如路由表），而用脏写接口去写非关键数据（例如用户的状态信息），读取数据用脏读接口。

Controller服务器向UMP集群提供各种管理服务，实现元数据存储、集群成员管理、MySQL实例管理、故障恢复、备份、迁移和扩容等功能。Controller服务器上运行了一组Mnesia分布式数据库服务，系统的元数据如集群成员、用户的配置和状态信息，以及用户名到后端MySQL实例地址的映射关系（路由表）等都存储在Mnesia里，其他服务器组件通过发送请求到Controller服务器获取用户数据。

为了达到高可用性，系统中会部署多台Controller服务器，它们通过ZooKeeper提供的分布式锁算法选举出一个leader，这个leader负责调度和监控各种系统任务，例如创建和删除数据库实例、备份和迁移等。这些系统任务可以分成多个步骤，而且会涉及系统中的多个组件，例如主库、从库和Proxy服务器等，还需要提供失败时回滚的方法。因此，我们采用类似工作流的方式来实现。每个系统任务都分成多个阶段的Erlang进程，每执行

完一个步骤跳进下个步骤之前会把中间状态持久化到Mnesia中。如果任务因为节点故障停止的话，leader能检测到并重新发起该任务，任务重启后会从上一次失败的“断点”继续向下执行。

API/Web服务器向用户提供了系统管理界面。它们是基于开源项目Mochiweb和Chicago Boss开发的，Mochiweb提供HTTP/HTTPS服务，而Chicago Boss是由Nginx的作者之一Evan Miller开发的，提供类似Rails的MVC框架。与Rails比，Erlang开发的框架天生就对并发有很好的支持，每个请求占用一个轻量级的Erlang进程，而Rails虽然在最近引入了多线程安全，但处理每条请求时仍然是独占整个进程的，因此需要使用多进程模型处理并发请求，通过Phusion Passenger等应用服务器进行派发。

Proxy服务器向用户提供访问MySQL数据库的服务，它完全实现了MySQL协议，用户可以使用已有的MySQL客户端连接到Proxy服务器，Proxy服务器通过用户名获取到用户的认证信息、资源配额的限制（例如最大连接数、QPS和IOPS等），以及后台MySQL实例的地址（列表），再将用户的SQL查询请求转发到正确的MySQL实例上。

除了数据路由的基本功能外，Proxy服务器中还实现了资源限制、屏蔽MySQL实例故障、读写分离、分库分表、记录用户访问日志等功能。Proxy服务器是无状态的，服务器宕机不会对系统中其他服务器造成影响，只会造成连接到该Proxy的用户连接断开。多台Proxy服务器采用LVS HA方案实现负载均衡，用户应用重连后会被LVS定向到其他的Proxy上。

Agent服务器部署在运行MySQL进程的机器上，用来管理每台物理机上的MySQL实例，执行创建、删除、备份、迁移和主从切换等操作，收集和分析MySQL进程的统计信息、bin log和slow query log。

日志分析服务器会存储和分析Proxy服务器传入的用户访问日志，并实现了实时索引供用户查询一段时间内的慢日志和统计报表。信息统计服务器定期将采集到的用户连接数、QPS数值，以及MySQL实例的进程状态用RRDtool进行统计，可画图展示到Web界面上，也可为今后实现弹性的资源分配和自动化的MySQL实例迁移提供依据。

UMP系统中各节点间的通信（不包括SQL查询、日志等大数据流的传输，这些还是直接走TCP的）都通过RabbitMQ，作为消息通信的中间件来使用，以保证消息发送的可靠性。ZooKeeper则主要发挥配置服务器、分布式锁，以及监控所有MySQL实例的作用。

在多个组件的协同作业下，整个系统实现了对用户透明的容灾、读写分离、分库分表功能。系统内部还通过多个小规模用户共享同一个MySQL实例，中等规模用户独占一个MySQL实例，多个MySQL实例共享同一个物理机的方式实现资源的虚拟化，降低整体成本。在资源隔离方面，通过Cgroup限制MySQL进程资源，以及在Proxy服务器端限制QPS相结合的方法，UMP系统能在实现资源虚拟化的同时保障用户的服务质量。此外，UMP系统综合运用SSL数据库连接、数据访问IP白名单、记录用户操作日志、SQL拦截等技术保护用户的数据安全。

结束语

UMP系统的一些组件，例如Proxy服务器和日志分析服务器，目前已经运用在天猫的聚石塔平台中，为电商和ISV提供安全的数据云服务。此外，UMP系统还运用在淘宝的店铺装修平台中，为开发者提供数据服务。下一阶段，我们希望UMP系统能进一步为企业降低数据存储的成本。



曹伟

花名鸣嵩，淘宝核心系统数据库组技术专家，目前从事高性能服务器、IM、P2P、微博等各类型分布式系统、海量存储产品的开发，关注系统的高可用性和一致性及分布式事务领域。

责任编辑：杨爽（yangshuang@csdn.net）

开源IaaS云平台的分析与比较

文 / 贾琨

作为云计算的一种重要形式，IaaS服务有各种开源和商业云平台方案。本文立足于使用开源IaaS云平台来开发公有云和私有云管理平台的角度，介绍和比较了Eucalyptus、OpenNebula、CloudStack和OpenStack等开源IaaS云平台。

从AWS看成功云平台的特点

AWS是当前最成功的云计算平台，其系统架构最大的特点就是通过Web Service接口开放数据和功能，一切以服务为第一位；并通过SOA的架构使系统达到松耦合。

AWS提供的Web Service栈，由访问层（API、管理控制台和各种命令行等），通用服务层（身份认证、监控、部署和自动化等），PaaS层服务（并行处理、内容传输和消息服务等），IaaS层服务（计算EC2、存储S3/EBS、网络服务VPC/ELB等以及数据库服务）几部分组成。用户应用使用IaaS基础IT资源，将PaaS和通用服务作为应用架构中的组件来构建自己的服务。综合来看，AWS生态环境中系统架构的核心思想为SOA、分层和服务组合。

私有云的需求

除了AWS这类公有云平台，私有云和混合云也是IaaS的重要形式。企业对于私有云平台通常会有以下几个需求。

- 计算虚拟技术的多样选择（KVM、XEN、ESX、ESXi、Hyper-V和XenServer等）。
- 存储技术/设备的多样支持（交换机、路由器和防火墙等）。
- 网络技术/设备的多种支持（NAS、IP-SAN和FC-SAN等）。
- 多种API的支持。

前三个需求要求IaaS平台能屏蔽底层的具体技术/设备的差别对外呈现基本一致的能力与接口。这一般要采用抽象框架加插件的设计来实现。另外，基于计算虚拟化、网络和存储等技术自成体系的原因，整个架构设计中须考虑将计算虚拟化、网络和存储独立成三个子系统或服务。

因此，云平台至少应包含三层：API或接入层提供各种不同API或访问方式，核心虚拟化管理层整合底层服务来对外提供IaaS服务，计算/存储/网络服务层屏蔽技术差异。

技术团队开发需求

小型技术团队精英化，每个人都能够参与整体设计。大型团队则为金字塔结构，只有少数人能够参与整体设计，多数人员因能力和职责的原因只能接触到单个功能或模块。

为满足这两种团队的要求，云平台的整体软件架构必须做到松耦合，通过组合组件、模块和服务来构成整个系统；同时需要组件、模块和服务功能内聚以便于小团队独立维护，方便独立的设计、开发和演进。

另外，云平台需要考虑提供基础共享组件在各个服务中重用。典型的可重用组件为数据库ORM、消息通信、服务端基础框架、配置管理系统、日志系统和错误定位系统等。很多大型团队会整合这

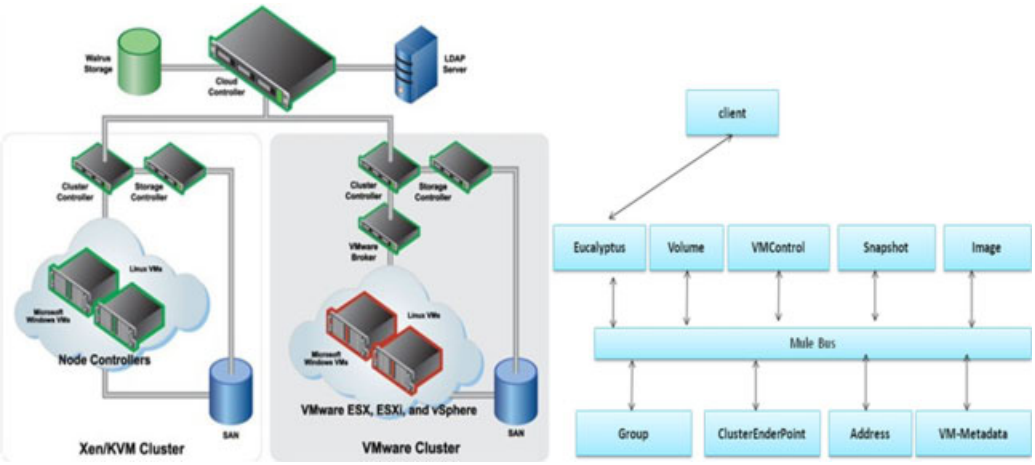


图1 Eucalyptus系统架构和CLC逻辑架构

些基础共享服务，通过领域描述语言自动化生成基础框架代码，使开发人员可以专注于具体的服务实现和关键技术研究。

云平台的介绍和比较

下面从系统架构要分层、组件化，采用SOA以达到系统松耦合；组件服务使用框架插件化设计；开发平台化等方面来比较4个开源IaaS云平台。

Eucalyptus

Eucalyptus是最早试图克隆AWS的开源IaaS云平台，整体架构如图1的左半部分所示。Eucalyptus由云控制器（CLC）、Walrus、集群控制器（CC）、存储控制器（SC）和节点控制器（NC）组成，它们相互协作共同提供所需的云服务。组件间使用支持WS-Security的SOAP消息实现安全的通信。Eucalyptus对外提供兼容AWS的SOAP和Query接口，不提供其他API。

从分层的角度来看，Eucalyptus缺乏API层设计，CLC是全局资源管理层，集群服务（CC和SC）为底层资源管理层。CLC、CC和NC三层结构不是软件架构层面的分层，只能看作一种为了管理较大规模集群的工程化方法。

从组件服务角度看，每个集群中将计算和存储服务设计为独立服务，网络仍为与计算服务的一部分。尽管Eucalyptus在代码实现上是将网络部分独立出来的，但整体上并未按照独立的服务来设计，整体设计解耦不够。

CLC是Eucalyptus的核心，包括虚拟机控制、存储卷管理、网络资源（Address）管理、镜像管理、快照管理、Keypair管理和元数据管理等服务模块。开源ESB Mule将所有的服务编排起来，通过Eucalyptus服务对外统一提供EC2和EBS服务，如图1的右半部分所示。由此可以看到，Eucalyptus在SOA层面上做得较好。但ESB技术门槛高，对设计开发人员要求较高。同时因为Eucalyptus只在很少的地方支持插件（如多Hypervisor支持的插件），所以整体上对抽象框架和插件的设计做得不多。

从开发平台的角度来看，Eucalyptus的主要开发语言为Java和C；CLC采用开源ESB Mule为核心编排服务，架构较新颖；但CC和NC采用Apache+CGI的软件架构，基于Axis/C来实现Web Service。整体来看，Eucalyptus还没有开发平台化的趋势。

OpenNebula

OpenNebula是Reservoir项目的一部分，是2005年欧洲研究学会发起的虚拟基础设备和云端运算计划的虚拟化管理层的开源实现。OpenNebula的核心部分是Front End，即ONE。其架构如图2所示。

OpenNebula明显分为三层，即接口层、核心层和驱动层。接口层提供了原生的XML-RPC接口，同时实现了EC2、OCCI和OpenNebula Cloud API（OCA）等多种API，为用户提供了多种选择。

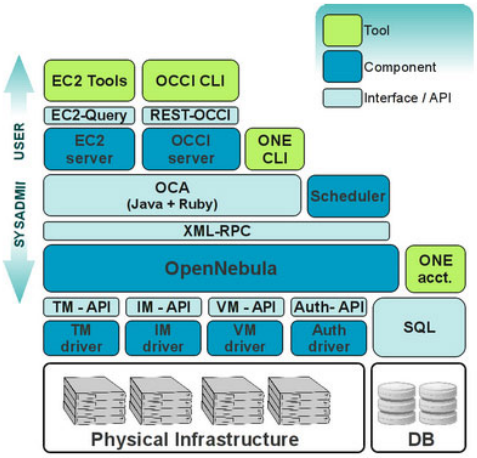


图2 OpenNebula系统架构

核心层的OpenNebula core提供统一的Hook插件管理、Request请求管理、VM生命周期管理、Hypervisor管理、网络资源管理和存储资源管理等核心功能。core配合Scheduler对外提供计算和存储网络资源管理服务。

最底层是由各种Driver构成的驱动层与虚拟化软件（KVM、XEN）和物理基础设施交互。需要说明的是，图2中的驱动层没有画出DataStore、NetworkManager等多个驱动。一些前端模块如监控、用户界面（Sunstone Portal和Self Service Portal）也未在图2中画出。很明显，OpenNebula在分层和框架加插件设计这两点做得很好。

在OpenNebula中，计算、存储和网络部分是ONE中独立的模块，资源调度也被分离出来通过requirement和matcher支持多种可选的策略和资源额度管理，也支持调度引擎Haizer来提供lease（租约）的高级资源调度能力。

显然，OpenNebula没有采用SOA的设计，没有将计算、存储和网络设计为独立组件，解耦做得还不够。值得注意的是，OpenNebula用Libvirt所提供的接口远程调用计算节点上的虚拟化控制命令。这种Agentless的设计在系统安装部署阶段会减少很多软件安装配置工作，是一个设计亮点。

从开发平台的角度来看，OpenNebula采用C++实现核心ONE，使用Ruby开发的各种Driver来实现具体的功能。整体系统只有一个核心部件，故在开发平台上做得很少。

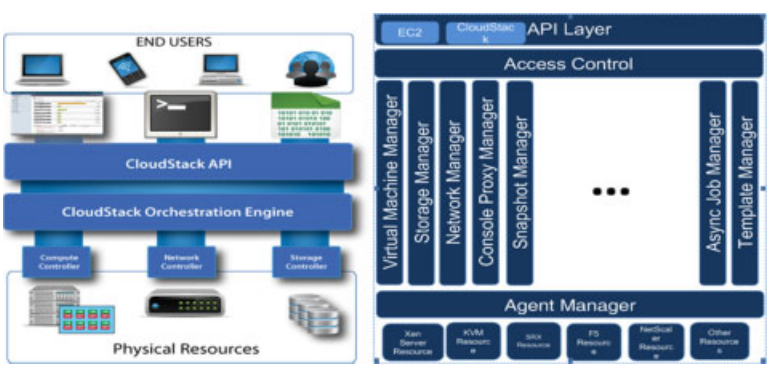


图3 CloudStack系统架构和Management Server架构

CloudStack

CloudStack是Cloud.com开发的开源IaaS软件，被Citrix收购后贡献给Apache基金会。它已为全球多个公有云提供IaaS平台技术，如英国电信（BT）、日本电报电话公司（NTT）和韩国电信（KT）等。

图3中的左半部分为CloudStack的总体架构，可以看到其包括Dashboard/CLI层、CLoudStack API、核心引擎层和计算/网络/存储控制器层，是典型的分层架构。具体来看，CloudStack提供原生自定义API，也通过cloud bridge支持AWS兼容API。

这两种API构成了API层。核心引擎层CloudStack Orchestration Engine（Management Server，如图3中的右半部分所示）是由Access Control、Async Job Manager和Job Execution等模块组成的抽象框架，它是一个业务编排引擎。VM Manager、Storage Manager和Snapshot Manager等模块通过调用底层资源控制器Agent来实现所有功能。

Management Server的最底层由Agent Manger及其管理的各种Hypervisor和网络设备Agent构成。这一层也很好地体现了框架加插件设计。

与OpenNebula类似，CloudStack本身也未采用SOA的设计，同样没有将计算/存储/网络部分从核心引擎中分离出来，因此在松耦合和组件设计上需要进一步加强。

从开发平台来看，ClousStack使用Java开发API、Management Server和Agent等部分，运行时部署为Tomcat的Serverlet。另外，还大量使用Python开发与网络和系统管理相关的部分。值得注意的是，CloudStack代码中包括一套独立的Java代码库，涵盖通信、数据管理、事件管理、任务管理和插件管理等部分，基本形成了开发平台。

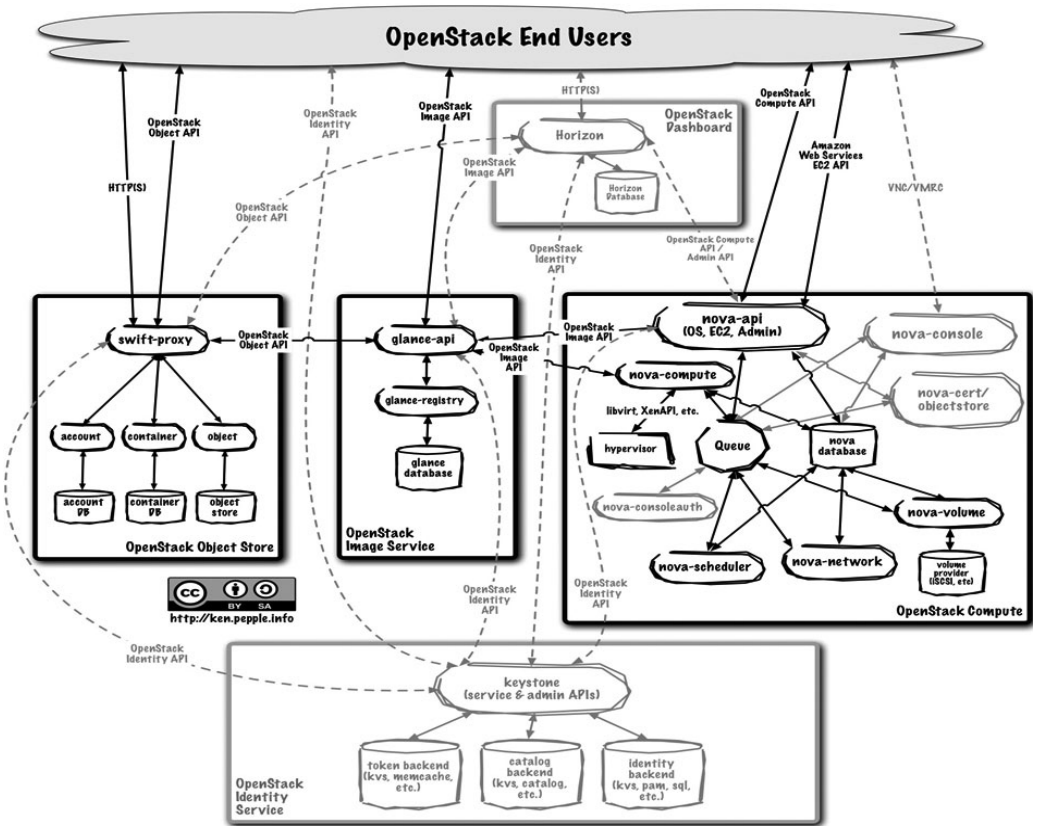


图4 OpenStack Essex逻辑架构

OpenStack

OpenStack是开源IaaS云平台的新兵，只有2年时间，却拥有最好的社区和生态环境，吸引了大量的公司和开发者围绕其进行云计算开发。图4为最新发布的Essex的逻辑架构图。

OpenStack整体架构分3层，最上层为应用程序和管理Portal（Horizon）、API等接入层；核心层包括计算服务（Nova）、存储服务（包括对象存储服务Swift和块存储服务cinder）和网络服务（Quantum）；第3层为共享服务，现在为账户权限管理服务（keystone）和镜像服务（Glance）。其中Quantum和cinder是最新加入核心服务中的OpenStack孵化项目。

在Essex及以前版本，存储EBS（Elastic Block Service，弹性块存储服务）和Nova-Volume耦合在一起，网络服务也与Nova-Network绑定。在正在开发的Folsom版本中，EBS和Network从Nova中独立为新的服务（cinder和Quantum）。Nova通过API来调用新的cinder和Quantum服务。我们可以看到，OpenStack在SOA和服务化组件解耦上是做得最好的。

Nova包含API Server（含CloudController）、Nova-Scheduler、Nova-Compute、Nova-Volume和Nova-Network等几部分，所有组件通过RabbitMQ来通信，使用数据库来保存数据。同时Nova中大量采用了框架与插件的设计，如Scheduler支持插件开发新的调度算法，Compute部分支持通过插件使用不同的Hypervisor，Network和Volume部分也通过插件支持不同厂商的技术和设备。cinder和Quantum等服务也采取了与Nova类似的整体架构和插件设计。

从开发平台的角度来看，OpenStack做得也很好。首先，OpenStack所有服务均采用Python开发；其次，所有服务采用类似的软件架构和内部实行技术，如服务端程序使用WSGI，数据库ORM使用SQLAlchemy，配置文件解析和日志等也采用相同的组件。基于OpenStack有很好的开发平台，我们看到开发人员可以很容易参与多个组件的开发。

表1 IaaS开源云平台比较

开源平台	分层架构	SOA	组件化/解耦	扩展性/插件设计	开发平台
Eucalyptus	较好 云、集群和节点分层，但缺乏API层	较好 CLC采用Mule编排服务；各服务间只使用Web Service接口交互	较好 计算和存储服务解耦；网络部分仍与计算虚拟化耦合	弱 只在虚拟化技术和调度算法上支持	弱 采用Mule作为服务总线
OpenNebula	好 分为API层、核心层和驱动层等	弱 只提供类EC2服务，无SOA设计	较好 整体一体化，通过Driver做到功能解耦	好 通过驱动层Driver实现	弱 没有开发平台
CloudStack	好 分为API层、核心编排层和资源层	弱 整体为一体化设计，无SOA设计	一般 整体一体化设计通过Agent来解耦	好 通过插件框架和Agent支持扩展	较好 初步形成开发平台，覆盖基本功能
OpenStack	好 整体分为Access层、核心服务层和共享服务层等。服务内仍有API、核心控制和底层资源服务等	好 完全按照SOA架构设计，服务间使用API接口	好 计算、网络、存储、镜像和身份认证等均为独立服务	好 各服务内部组件均支持框架和插件设计	好 所有服务采用统一开发语言、类似的软件架构和实现技术。除此之外，还明确提出设计原则来指导设计开发

综合比较

前面分别介绍了各IaaS开源云平台在分层、SOA、组件化、解耦及开发平台等方面的情况。

从表1的对比中可以看出，所有的开源IaaS云平台在分层上做得都比较好；在SOA/组件化/解耦这点上来看，OpenStack和Eucalyptus有优势；在框架和插件设计上，除Eucalyptus较差外，其他平台均有很好的设计——OpenStack的开发平台做得最好，CloudStack次之。综合来看，目前OpenStack的设计是最好的，Eucalyptus和CloudStack次之。

实际需求设计比较

让我们用一个真实需求来看4个开源IaaS平台在开发支持上的表现。此需求来自私有云场景，云平台需要对不同用户的资源请求（如VM和公网IP等）按优先级排序后进行处理，并提供任务的管理功能，如统计各状态的任务数量等。

需求的设计有两个关键点：一为如何对任务进行统一调度管理，二为任务状态转变信息的收集。

任务的统一调度管理方案分别为：OpenNebula和OpenStack都提供独立的Scheduler组件并支持扩展Scheduler的插件机制；CloudStack有Job Manager但不提供扩展，需修改Job Manager核心代码；Eucalyptus内部流程主要由Mule总线来驱动，需修改核心流程代码增加新的模块。比较来看，OpenStack和OpenNebula的实现方式对现有系统影响最小。

对于任务状态转变信息，由于信息遍布在系统多个地方，最佳的设计是通过消息发送状态变化给负责任务管理/统计的模块统一处理。在这一点上采用Message Bus的OpenStack和采用Mule的Eucalyptus有明显优势。综合来看，OpenStack为二次开发提供了很好的支持。

技术之外

上述比较主要是在设计方面，OpenStack优势显著。但从其他方面来看：

- Eucalyptus由于出现最早，同时与AWS签订相关API兼容协议，在面向AWS生态环境的私有云市场处于领先地位；
- CloudStack在经过大量商业客户公有云的部署后，其功能已趋于稳定成熟，成为Apache开源项目后，其松耦合设计也已排上日程，设计上大有迎头赶上的趋势；
- OpenStack现状是功能不够完整且商业支持不够，另其转为基金会运作后能否保持现在的发展趋势也是大家非常关注的。在实际的云平台选择过程中，大家要从自身的角度出发综合考虑功能和系统的架构与设计、未来发展等。



贾琨

北京天云融创科技有限公司云平台开发工程师和架构师。从2005年起从事Web、分布式、网格、HPC和云计算系统开发。精通资源调度管理和分布式计算技术。

责任编辑：杨爽（yangshuang@csdn.net）

Trident Storm与流计算经验

文 / 金晓军

在整体架构上，Storm与Hadoop非常相似，且弥补了Hadoop实时性差的缺点，因此被称为“实时的Hadoop系统”。本文结合实例深入讲述了Storm的Trident特性，同时分享了作者多年积累的流计算经验。

Storm

“实时的Hadoop系统”

随着互联网业务对数据处理实时性的要求越来越高，原本承载着海量数据处理任务的Hadoop在实时计算方面显得越来越乏力。在整体架构上，Storm与Hadoop非常相似，弥补了Hadoop实时性差的缺点，并且从系统本身解决了流计算中常见的容错、保序等问题。因此，被称为“实时的Hadoop系统”。表1是Hadoop和Storm的对比。

基本元语强大的表达能力

MapReduce将海量数据的并行计算逻辑，拆分成Map和Reduce两个过程。用户使用MapReduce处理业务逻辑时，需要将计算问题进行拆解，用“分治思想”来求解问题，然后套用MapReduce模型即可。

而对于Storm而言，表达用户业务逻辑就要简单很多。用户将业务数据流转化为Storm的Topology，数据源头设计为Spout，中间处理环节设计为Bolt，在需要满足保序或分流的地方，根据需求定制合适的Grouping策略（如图1所示）。

在需要做到数据不丢失时，使用基于xor逻辑的Acker来保证数据至少被正确处理一次。当需要更复杂的容错或保序处理时，可以使用Storm最基本的Spout、Bolt和Acker元语进行封装。

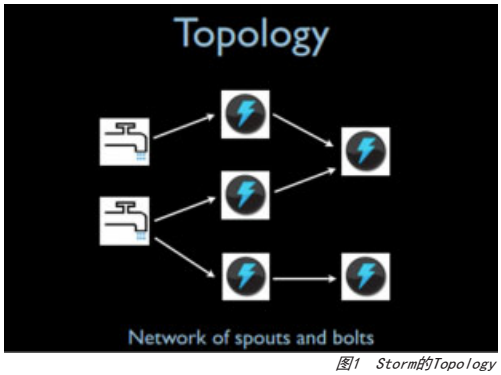


图1 Storm的Topology

基本元语的易用性及基于基本元语的抽象

使用过Storm的人都遇到过以下问题。

- 因为代码的编写不当，导致整个Topology没有像预想中的那样——数据至少被处理一次。
- 业务逻辑是使用SQL描述的，而在实现这个逻辑时却需要转为成由Spout和Bolt组成的Topology。
- 在使用Storm的过程中，需要将中间结果或最终结果输出到NoSQL或其他数据库中，同时还需要保证数据的一致性。

有没有一种方式能从系统层面解决这些问题呢?

表1 Hadoop vs. Storm

	Hadoop	Storm	作用
系统角色	JobTracker	Nimbus	任务调度，资源管理
	TaskTracker	Supervisor	启停执行进程，汇报节点状态
	Child	Worker	业务逻辑插件加载进程
应用名称	Job	Topology	用户自定义的任务
组件接口	Mapper/Reducer	Spout/Bolt	编程模型

与MapReduce类似，Storm基本元语的表达能力是非常强大的，但其在某一类特定场景的易用性，却做得不是很好。一个比较好的做法是：针对某一类特定场景，进行适当的抽象，并构造出更加贴近业务、更易用的基本API或元语。

Cascading在Hadoop之上封装了更加丰富的API，使你不需要接触MapReduce任务就能使用分布式计算能力；Mahout在Hadoop之上实现了各种机器学习和数据挖掘算法库，可以更方便地提供推荐服务；而Hive和Pig为用户提供了更方便的类似于SQL的接口，通过SQL就可以使用分布式计算。

与Hadoop体系类似，Twitter的Nathan Marz在Storm中提供了三类（DRPC、Transactional Topology和 Trident）更易用的高级元语。Distributed-RPC基于Storm的基本元语，实现了一个高度容错的分布式同步RPC框架；Transactional Topology也是基于Storm的基本元语，使Storm做到每一个Tuple被且仅被处理一次；而Trident则比Transactional Topology的能力更进了一步，它不仅提供exactly-once的语义，同时提供了由Partition-local、Repartitioning、Aggregation、Group和Merges and Joins五类基本元语组成的高层抽象系统。

Trident

Trident是什么?

Trident是Storm 0.8.0版本引入的新特性，为基于Storm元语进行实时计算的用户提供了一类更高级的抽象元语，能够同时满足高吞吐量（每秒百万级的消息）与低处理延时。Trident提供了Partition-local、Repartitioning、Aggregation、Group和Merges and Joins这五类基本元语，同时延续了Transactional Topology的exactly-once语义，能够满足用户一些事务性的需求，但前提条件是对性能要有所牺牲。

下面将结合一个例子来简单介绍一下Trident的功能。例子中要完成两个功能：

- 统计WordCount；
- 通过执行查询得到一组单词的总数。

以下代码是模拟输入的Spout代码：

```
FixedBatchSpout spout = new FixedBatchSpout(
  new Fields("sentence"), 3,
  new Values("the cow jumped over the moon"),
  new Values("the man went to the store\
    and bought some candy"),
  new Values("four score and seven years ago"),
  new Values("how many apples can you eat"))
  spout.setCycle(true);
//true表示会循环的产生以上values
```

如果我们要用Storm的基本元语来实现这个功能应该怎么做呢?

```
// 创建一个TopologyBuilder
TopologyBuilder builder = new TopologyBuilder();
// 设置spout，component id设为sentences
builder.setSpout("sentences", spout, 5);
// 设置分句的bolt，上游component为sentences
builder.setBolt("split", new SplitSentence(), 8)
.shuffleGrouping("sentences");
// 设置统计word count的bolt，上游component
为split，grouping策略是按word做hash
builder.setBolt("count", new WordCount(), 12)
.fieldsGrouping("split", new Fields("word"));

// 再创建一个DRPC client，但外部查询调用
DRPCClient client = new DRPCClient(
  "drpc.server.location",port);
/*
  -//定义DRPC的代码
*/
```

其中，SplitSentence与WordCount是两个Bolt类。

而对于Trident来说，如何完成这个功能呢?

一种实现是，只通过Trident进行计算过程的描述，再单独创建一个DRPC client供查询调用：

```
TridentTopology topology = new TridentTopology();
TridentState wordCounts =
  topology.newStream("spout1", spout)
  .each(
    new Fields("sentence"),
    new Split(),
    new Fields("word"))
  .groupBy(new Fields("word"))
  .persistentAggregate(
    new MemoryMapState.Factory(),
    new Count(),
    new Fields("count"))
  .parallelismHint(6);
// 再创建一个DRPC client，但外部查询调用
DRPCClient client = new DRPCClient(
  "drpc.server.location", port);
/*
  -//定义DRPC的代码
*/
```

另一种实现是直接通过Trident的Topology创建一个DRPCStream：

```
topology.newDRPCStream("words")
.each(new Fields("args"),
      new Split(),
      new Fields("word"))
.groupBy(new Fields("word"))
.stateQuery(wordCounts,
            new Fields("word"),
            new MapGet(),
            new Fields("count"))
.each(new Fields("count"),
      new FilterNull())
.aggregate(new Fields("count"),
           new Sum(),
           new Fields("sum"));
```

而在上面的例子中，Split、Count和Sum是简单的Function，而不再是Bolt。可以看出，通过

Trident这层更高级的抽象层，用户再也看不到Bolt、Ack、Emit和BatchBolt等概念，也不用去想Execute和finishBatch里该写什么；根据业务需求的数据处理逻辑，对Stream进行简单的groupBy、Aggregate操作即可。

当然，以上只是Trident从API层面带来的最直观的优势，Trident更强大的地方将在下文逐一介绍。

Trident中的基本概念

对于Trident来说，用户所要关心的对象只有Stream，Trident的所有API也都是针对Stream的。

因此，基本元语也不再是Storm的Spout、Bolt和Acker等，而是五类新的元语。

- Partition：不需要网络传输，在本地就能完成的操作。
- Repartition：决定Tuple如何分发到下一个处理环节的操作。
- Aggregation：Partition处理结果的汇聚操作。
- groupBy：对Stream中的Tuple进行重新分组，后续的操作将针对每一个分组独立进行，类似于SQL里的groupBy。
- Merges and Joins：将多个Stream融合为一个Stream。

Trident如何保证处理过程的一致性

Trident对进行读写资源的操作做了非常好的抽象工作，将这些有状态的操作全部融入到TridentTopology的内部，用户可以帮Trident将运行过程中间状态（State）写到HBase、MySQL等相关NoSQL或其他数据库产品中。这些操作对于Trident的API来说，是没有任何区别的。

Trident通过幂等形式对中间状态进行管理，做到即使发生重试或者操作处理节点挂掉的情况，也能保证每一个Tuple会被处理一次（exactly-once语义）。

Trident定义了三个基本的语义，可以实现exactly-once的以下功能。

- 所有的Tuple是以小批量的形式进行处理的，而不是Storm基本功能中的单条数据处理。
- 在每一批数据系统处理的过程中，会生成一个严格递增的事务id（txid）。如果这批数据因为处理超时或者没有正确处理完，则这批数据将会被重发，重发后txid保持不变。
- 不同批次数据的计算是乱序执行的，但中间状态的提交是按照txid的顺序进行的。上一批数据的计算结果没有被正确提交，下一批数据的结果是不会被提交的。

当然，要做到exactly-once还需要源头Spout的配合。根据对不同容错级别的定义，Trident将Spout分为三类：non-transactional Spout、transactional Spout和opaque transactional Spout。

non-transactional Spout

顾名思义，Spout对每一批数据的容错处理不做任何保证，使用这类Spout无法做到exactly-once。如果一批数据处理失败不重发的话，那么只能做到数据至多被处理一次；如果一批数据处理失败重发的话，那么只能做到数据至少被处理一次，但有可能出现同一条数据被处理多次的情况。

transactional Spout

事务性的Spout，使用这类Spout，用户可以轻易做到exactly-once，但这类Spout对消息队列有以下三个基本要求。

- Trident会为每批数据分配一个txid，给定一个txid，得到的批次是相同的，消息队列重发过来的数据必须与第一次来的数据有相同的txid。
- 不同批的数据之间不能有重叠部分。
- 每个Tuple必须置于一个批次中，不能漏数据。

使用transactional Spout时，用户维护中间状态的代码会变得非常简单，并且可以做到只处理一次，非常适用于counter场景。

举一个WordCount的例子。

数据库中当时的状态为：

```
man => [count=3, txid=1]
dog => [count=4, txid=3]
apple => [count=10, txid=2]
```

现在来了一批数据，txid=3

```
["man"]
["man"]
["dog"]
```

对“man”来说，最后一次来的txid=1，txid=3的数据需要处理。而对“dog”来说，最后一次来的txid=3，所以txid=3的数据已经处理过，故这批是重发过来的，不做处理。最后得到的结果是：

```
man => [count=5, txid=3]
dog => [count=4, txid=3]
apple => [count=10, txid=2]
```

如果你所使用消息队列无法保证在系统可以根据指定的txid取得同一批的数据，那么就无法满足transactional Spout第一个基本要求，transactional Spout就无能为力了。

opaque transactional Spout

与transactional Spout相比，opaque transactional Spout对消息队列的要求要宽松一些：每个Tuple都会在一个批次的数据中。如果Tuple在某一批处理失败，那么可以将这个Tuple重发到后续的批次中；不同批数据不能有重叠：不可以漏数据。

transactional Spout是透明的，用户只需记录中间结果和最后一次成功处理的txid，就能完成exactly-once语义。而opaque transactional Spout对消息队列的要求要宽松一些，但对用户的操作来说，无法做到透明。用户需要记录一些历史结果，配合Trident完成exactly-once的语义。

假设一个word的当时状态如下：

```
{
  value = 23,    // 当时统计的值
  prevValue = 1,// 在txid这批来之前的值
  txid = 2,      // 最后一次处理过的批txid
}
```

现在来了一批数据，word的统计值是20，txid=3。

因为最新的txid是3，大于最后一次处理过的txid，所以是新来的数据。最终结果值为原来的value加上这批新来的值，prevValue设为原来的value，最后一次处理过的txid设为3，最终的结果为：

```
{
  value = 43,
  prevValue = 23,
  txid = 3,
}
```

但如果现在来的这批数据，txid=2，我们又将进行怎样的逻辑处理呢？

因为最后一次处理过的txid也等于2，说明在处理txid=2这批数据时，整个Stream没有被处理完，所以这批数据是重发的。这就意味着要重新处理，因此新的value应该为prevValue+20，最终的结果为：

```
{
  value = 21,
  prevValue = 1,
  txid = 2,
}
```

这时有个疑问，在第一次计算txid=2这批数据时，得到的结果是23，而新的这批数据计算结果却为21，少了的那两条数据是丢了吗？

这两条数据并没有丢失。每个Tuple都会在一个批次的数据中，如果Tuple在某一批处理失败，可以将这个Tuple重发到后续的批次中。这两条数据将在后续的批次中重发，如果像处理transactional Spout那样，txid=2的这批数据直接不处理，就会导致多次计算。

根据上述对三类Spout的比较，可得出如下结论。

- non-transactional Spout无法做到exactly-once；
- transactional Spout可以做到exactly-once，且对于用户来说，使用上是透明的，不需要关注值的中间状态，但对数据源头的要求比较苛刻。
- opaque transactional Spout也可以做到达exactly-once，且功能是最强的。但对于用户的使用上，需要精心地设计中间状态的存储。对数据源头的要求相对要低一些。

流计算的经验总结

流计算的特点

- 实时性：这是流计算与批量计算最大的区别。要做到这点，就需要数据的切片尽可能小，数据推送更加及时，像Hadoop那样采用HDFS或HBase切片数据推送的方式已很难满足需求，这时需要更加高效且容错做得很好的消息中间件。
- 不确定性：要处理的Stream是由业务产生的有向无界数据，在某些方面是不可控的。数据的到达时机不可控，数据质量不可控，上游数据（业务渠道）不可控。因此，这对整个系统在容错与可扩展性方面有非常高的要求。
- 无终止状态：对流计算而言，数据流源源不断，很难找到一个明确的状态分界线，因此，流计算的计算结果是一系统连续的中间状态，下游数据

对这一系统的中间状态进行加工和使用。而对于“整合计算”而言，需要对批量计算的结果与流计算的中间状态进行整合。

而流计算系统，就是要根据流计算的特点去解决流计算特点带来的问题。Storm通过非常灵巧的方式，解决了上述的问题。

Storm的定位

很多人在使用Storm时都存在这样的误区。

1. 业务流程非常简单，只需要一个Spout和一个Bolt就能搞定，有必要用Storm吗?

2. 整个集群只跑了1个Topology，有没有必要用Storm?

3. 数据量很小，单机就能搞定的需求，有必要用Storm吗?

4. 按小时维度做统计的结果，要用Storm吗?

任何系统都有自己的适用场景，Storm也不例外。在不合适的场景，滥用Storm，不仅使整个系统的架构变得复杂，而且更增加了对Storm的学习成本、运维成本和监控成本等。

对于1，采用消息中间件+处理器比较合适。

对于2，Storm是一套支持多任务的分布式计算系统，只运行1个Topology未必能将资源跑满，会造成资源上的浪费。

对于3，使用Storm不合适，一个单机程序即可。

对于4，Storm解决的是对实时性要求非常高的计算需求，对于小时级的需要，用Hadoop更合适。

Storm存在的问题

Storm在解决流计算问题时，提供了一个非常好框架，但也存在以下一些问题。

- Nimbus的单点问题：对Supervisor和Worker来说，Storm有很好的容错策略。但对Nimbus来说，却没有备份方案。如果Nimbus所在的机器宕机，那么之前用户提交的Topology信息，便可能丢失。同时，新的Topology也无法提交，对需要持续提供服务的平台，这个问题的出现是灾难性的。

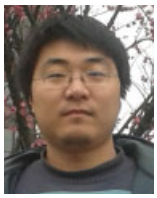
- 监控系统不够完善：Storm UI只记录了每个组件处理的数据量。无法做到对某个Tuple精确定位。

- Storm的精简其实是个伪命题：Storm的内核确实很精简，并且对编程模型进行了很好的抽象。但它把“脏活累活”抛给了外围系统，抛给了用户。如果需要做到记录级容错，需要使用具有做到容错功能的数据源头，但Storm没有解决数据源头容错的问题。同样，Storm没有解决中间状态持久化的问题，也需要依赖外部系统或者用户自己去维护。

- Storm调度系统的问题：尽管在0.8.0版本中引入pluggable scheduler去试图解决调度的问题。但这也仅仅是只提供了接口，没有提供一套根据当前集群、不同节点的状态与负载情况，进行Worker的rebalance。同时，也没有像Mesos那样，对Worker进行物理上的隔离，如果一个Topology出问题，可能会影响到其他Topology。

总结

Storm并不解决所有流计算问题的银弹，技术方案选型，要结合自己的业务场景。只有理解Storm背后的设计思想，弄清Storm底层的实现原理，才能在使用Storm时得心应手，解决实时计算问题时游刃有余。



金晓军

花名仙隐，就职于阿里巴巴集团数据平台部，负责流计算引擎和平台的研发等相关工作。一直致力于Hadoop、分布式计算、实时计算和搜索技术领域的研究工作。

责任编辑：杨爽（yangshuang@csdn.net）

腾讯分析系统架构解析

文 / 楚大鹏，王琦英

TA（Tencent Analytics，腾讯分析）是一款面向第三方站长的免费网站分析系统，在数据稳定性、及时性方面广受站长好评，其秒级的实时数据更新频率也获得业界的认可。本文将从实时数据处理、数据存储等多个方面带你深入探寻TA的系统架构及实现原理。

网站分析（Web Analytics）主要指的是基于网站的用户浏览行为，对网站的点击流数据和运营数据进行分析，以监控网站的运营状况，为网站的优化提供决策依据。网站分析系统已成为站长日常运营必不可少的工具，业界比较流行的网站分析系统主要有Google Analytics、CNZZ和百度统计等产品。

TA作为网站分析产品的后起之秀在社区分析、用户画像、网站工具等多方面形成了自己的特色，其秒级的实时数据更新频率更是业界翘楚。在数据稳定性、准确性和及时性方面，TA在站长圈也是享有良好的口碑。

随着接入业务量的不断发展，TA日均需要处理和计算的数据量达到TB级。如此庞大的数据量想要达到秒级实时且保证系统的高可用并非件易事。

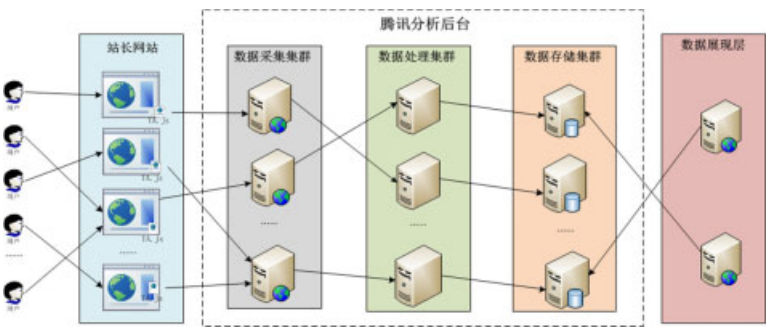


图1　TA的基本原理

TA的实时计算框架借鉴了一些业界流行的流式计算系统的思路。虽然在构建系统中遇到了一些问题，但由于海量数据的实时处理、实时存储具备一定的典型性与通用性，所以将TA的解决方案分享出来，希望能给大家一些启示。

基本原理及系统架构

TA的基本原理是通过嵌入站长网站的JavaScript脚本收集用户访问行为数据，并发送TA采集集群，采集集群收到数据后将其过滤、编码、格式化后继续向后分发。数据处理集群负责按照业务逻辑计算数据，并将计算结果“写入”到数据存储集群，最后将结果数据展现给广大站长使用。TA的基本原理如图1所示。

TA后台是一套完整的数据流处理系统：由JavaScript采集的用户行为数据像川流不息的河水一样流入TA后台，经过清洗、计算后源源不断地流出到TA存储集群，供用户浏览和查询。TA的具体架构及核心部件如图2所示。

TA的后台分为离线和实时两部分：实时部分负责系统的主要功能计算，数据更新频率为秒级；离线部分负责系统复杂的关联分析及跨天计算，数据更新频率为天级。

- Http Access：主要负责HTTP协议的解析，数据的清洗及格式化。
- ESC：Event Streaming Coder，主要负责将系统不可枚举的数据类型编码成为整型，并将对应关系持久化。
- ESP：Event Streaming Processor，主要负责将数据按照站点、UID重新组织并计算PV、UV、停留时长和蹦失率等网站分析指标。
- ESA：Event Streaming Aggregator，主要负责汇总ESP计算后的数据按照站点，并写入到Redis。
- Center：系统的中心节点，负责系统配置、数据路由管理，并承担容灾切换功能。
- Logserver：负责将Access收集到的数据以字符串形式写入文件，并上传到TDCP上。
- TDCP：腾讯分布式计算平台，负责离线数据的计算，并由脚本将结果数据写入MySQL中。

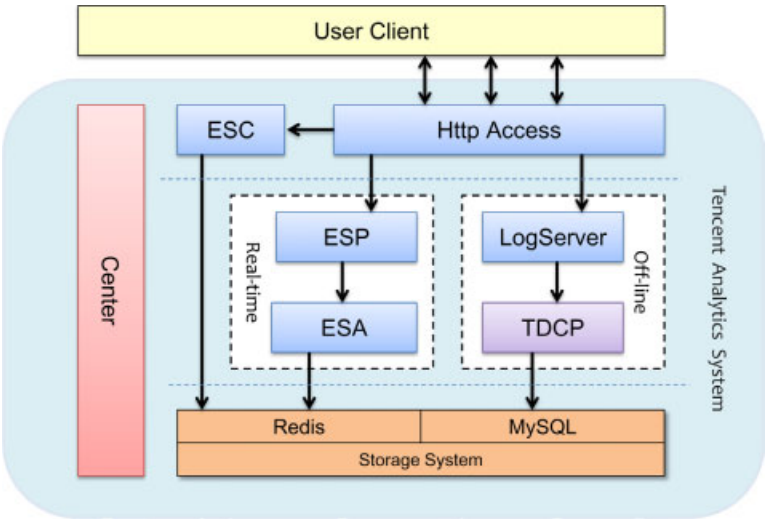


图2 TA的具体架构及核心部件

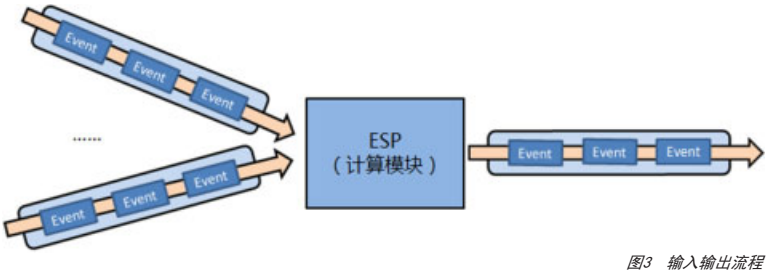


图3 输入输出流程

实时解决方案

在介绍TA实时解决方案前，我们先来了解下TA支撑的业务量。当前TA日均需要处理几十万网站的上TB级数据，处理过后的URL个数仍有上亿条，系统存储的key个数超过十亿。如何高效、低延迟地处理如此大量的业务数据是TA实时系统面临的主要挑战。TA解决方案的主要思路可以概括为数据全二进制化、计算全内存化、存储NoSQL化。下面就实时计算和实时存储这两大子系统进行深入的讨论。

实时计算

对于计算子系统，我们参考了Hadoop、S4和Storm等开源项目，力图设计为一个较为通用，扩展性较强的全内存实时Event处理系统（或者套用流行的术语称为流式实时Event处理系统）。对于这样的一个系统，我们设计支持的典型输入输出流程大致如图3所示。

实时计算系统的设计要点在数据组织、协议和增量计算模型上。

数据组织。万物皆int，考虑到内存以及计算过程的性能需求，我们将所有非int的数据类型转化为int。可以枚举的数据类型，将其配置化映射为唯一int；不可枚举的数据类型，则利用MD5算法近似得到唯一的int。例如，页面URL属于不可枚举的类型，则预处理通过MD5算法近似得到唯一的int；UserAgent里的浏览器类型字符串则属于可枚举的数据，则预先配置化映射为int。这个方法节省了较多内存，提高了整个系统的计算性能。

协议。协议层面上，我们首先设计实现了一种可扩展的Event结构，这种Event结构支持半自动化的序列化/反序列化机制（参考自msgpack的设计）和紧凑的二进制编码（基于Zigzag编码，参考Protobuf的实现）。这种Event结构在流式高性能I/O（网络传输和持久化）方面表现得相当良好。实时计算子系统被设计为可以扩展支持任意的Event实现。

增量计算模型。增量计算模型，指的是基本计算过程，被定义为以下三部分（如图4所示）。

- Processor：负责具体业务逻辑的计算处理。
- Data Holder：负责保存增量结果数据，以及计算依赖的中间状态数据。
- Emitter：负责定期输出清空增量计算结果。

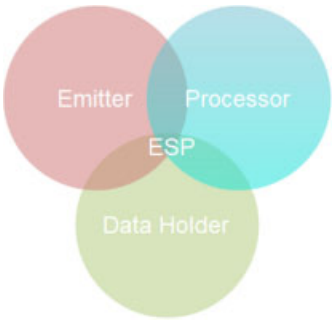


图4 增量计算模型的3个组成部分

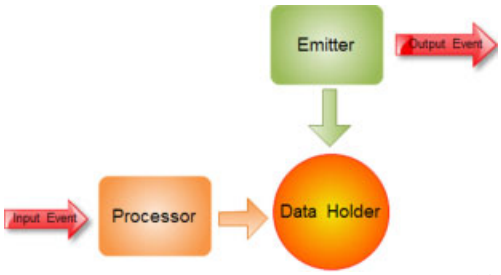


图5 处理流程

具体到流程方面，分为以下三步（如图5所示）。

- 接收Event，计算处理——Processor。
- 保存计算结果以及计算依赖中间数据——Data Holder。
- 定时触发输出时间片内计算结果，清空计算结果——Emitter。

增量计算模型弱化了分布式系统中单台机器的事务状态，相应地简化了分布式计算系统的实现，同时也提高了整个系统的性能。

实时存储

在TA系统中，实时存储的数据都是需要通过Web展示层读取的统计数据。这类数据存在两个典型特点。

- 频繁更新写。更新频度视系统实时性而定，每条统计结果更新频度最快可以达到1秒。
- 少量读取。“少量”是相对上述更新而言的。

同时根据业务逻辑，可将统计数据划分为两类。

- 固定不变数据：主要是URL、搜索关键词等数据。这一部分数据理论上是在不停地增加，不会修改旧有数据。
- 动态数据：主要是频繁更新的结果统计数据。这一部分数据则需要不停地更新。例如，www.qq.com域名下的PV和UV统计结果。

考虑到上述的TA实时统计数据的特点，我们选择NoSQL实现我们的存储系统；同时，针对两类不同的数据类型，分别选用LevelDB和Redis来存储。

Redis

TA实时存储的主要构件。考虑到TA系统本身就是一个比较完善的分布式集群系统，因此我们需要的存储构件是“not clustering, but sharding”。也就是说像HBase和MongoDB这样的“重武器”并不适合TA，而NoSQL数据库中的“瑞士军刀”Redis凭借其出色的性能走入我们的视野。同时TA的结果数据类型也比较丰富，有像站点PV、UV、VV和IP等Hash类型的数据，也有像用户访问轨迹这样set类型的“动态数据”，而Redis丰富的数据结构很好地完成了这项任务。

选择Redis的另一个原因是它足够简单且易于扩展。在实际应用的过程中，我们发现的问题都可以通过扩展Redis命令来解决。

例如，TA中有这样的一种应用场景：为了消除ESA模块的状态，存储在Redis中的数据往往并不是最终的结果数据，而是还需要进一步运算的中间数据。像bounce rate这个指标（bounce rate=bounce session数/total session数），需要前台查询两次再做一次运算后最终展示给用户。在高并发的情况下，无疑会影响系统的响应速度。

本着“移动计算，而不是移动数据”的原则，我们对Redis的sort、hmget命令进行了扩展使其支持四则运算，成功地将原来的两次查询优化为一次。扩展四则运算的另外一个目的是可以“通过计算换取存储”，例如需要将两种类型加总成总和的类型数据，可以只存储两份，加总数据“通过计算换取”。

除了数据读取，数据的写入也可以进行类似合并数据的优化。例如，TA在写入URL的PV、UV、VV、IP、停留时长和bounce rate这6个指标时，需要调用6次Redis命令。而实际上这6个指标是存储在同一个Hash内的，通过扩展hmincrby命令，支持将Hash的所有field一次更改，便能将调用次数优化至一次。上线之后也取得了良好的效果，峰

值时的CPU利用率几乎下降了一半，同时也大幅提升了上层模块ESA的吞吐量。

LevelDB

它是Redis的有效补充。考虑到Redis为内存数据库，而使用内存的成本要高于硬盘，因此选择引入了基于磁盘存储的LevelDB作为补充。由于LevelDB的写性能足够好，而读性能也远远超过目前“在线少量读取”的需求，所以我们选择LevelDB存储“固定不变数据”。

在数据存储的架构设计上，由于实时数据服务与在线系统，可靠性要求较高，因此我们主要采取双写复制+Sharding的设计方法。

- **双写复制**。所有的数据存储都会至少同步写两份，以提高在线系统服务的可用性。
- **数据分片**（Sharding）。

基于域名：所有的数据以域名为单位组织分片；任何域名可以调整到任意分片中；单个域名数据原则上存储在一个分片中。

动态调整（如图6所示）：只调整分片策略，不移动数据；基于数据量计算分片负载。

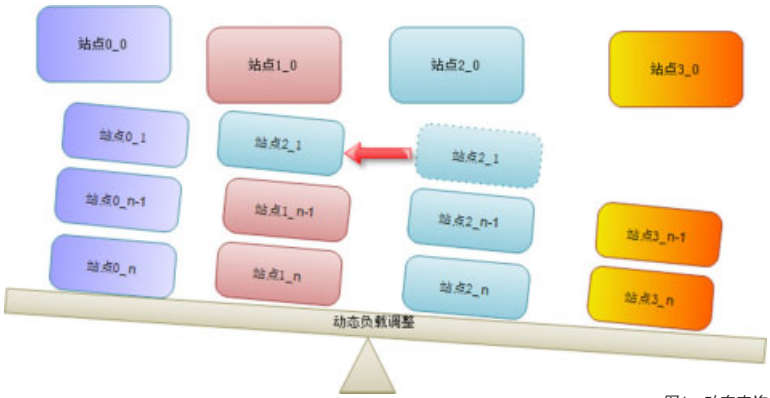


图6 动态查询

此外，针对分片集群数据的查询，我们主要做了三项工作（如图7所示）。

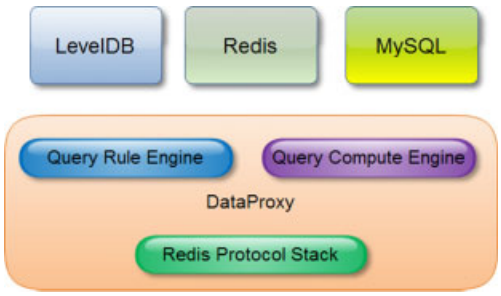


图7 应对分片集群数据查询

- Redis Protocol Stack是一个较为完整的Redis协议栈，是上层应用的基础。直接用Redis协议作为对外提供查询的通用协议，这样外部用户可直接通过目前各种Redis Client实现来查询访问数据。
- Query Rule Engine是一个灵活的查询引擎。能够根据规则智能地在多个Redis、LevelDB数据源中查询，执行类join的操作；也简单扩展支持其他的异构数据源，如MySQL、HBase等。
- Query Compute Engine是一个实时查询计算引擎，能根据基础查询结果实时计算。引入此部分的主要目的在于减少Redis数据空间占用。

未来展望

目前TA虽然在后台上已经做到数据秒级更新，但展示方式仍为传统的静态方式。后续TA会在数据的动态刷新上进行更多尝试，让站长可以第一时间了解网站营销效果，时刻感受网站心跳。



楚大鹏

腾讯数据平台部高级工程师，腾讯分析技术负责人，在网站分析、流式数据处理、海量数据挖掘等领域方面经验丰富。



王琦英

腾讯数据平台部高级工程师，负责腾讯分析的架构设计及开发，有丰富的高并发、高性能后台系统设计、架构经验。

责任编辑：杨爽（yangshuang@csdn.net）

HipHop算法
利用微博互动关系挖掘社交圈

文 / 张俊林

HipHop算法成功弥补了原有社交圈挖掘算法的不足，能更精准地挖掘微博互动关系中有价值的信息。本文讲述了HipHop算法出现的缘由及价值，同时结合示例给出了算法的实现步骤。

在微博环境下，如何自动挖掘某个用户的社交圈或兴趣圈是个很基础但很重要的问题。准确挖掘某个用户在微博上体现的社交关系，对很多具体应用来说都很有价值。例如，可以更好地对用户的兴趣进行挖掘或者能够推荐用户还未关注的社交圈子成员等，或者根据其社交圈子更准确地对用户进行个性化建模，为其他基于用户个性化模型的推荐或者广告推送等提供基础服务。

我们在微博相关研发任务中提出了HipHop算法，旨在通过利用微博用户的互动行为，来自动挖掘出用户的不同社交圈子。在设计算法之初，我们希望圈子挖掘算法能同时满足以下几个条件。

- 对于某个微博用户A来说，可以挖掘出其所属的多种社交圈子，例如用户既有的同事关系圈、所属的专业兴趣圈等。
- 同时对于用户B来说，他可能同时属于用户A的不同社交圈。比如B既是A的大学同学，也是A的某公司同事，那么B应该同时出现在用户A的两个不同兴趣圈里。
- 不使用用户隐私数据。出于保护用户隐私的目的，我们希望算法只使用用户的公开行为和信息，因此HipHop算法只使用了互动关系这种公众完全可见的公开信息。
- 社交圈可解释，即可以通过简洁的方式描述社交圈的性质或者特点。目前是通过给每个圈子打上不同的标签来进行区分。

HipHop社交圈挖掘算法就是根据这几个指导原则设计开发出来的，它能够同时满足以上几个约束条件。目前公开的参考文献中很少见到能够同时满足这些条件的相关社交圈挖掘算法。

常见的社交圈挖掘算法

社交圈挖掘是目前社交网络研究中非常典型和热门的研究任务，通常被称为“社群发现”。学术界也陆续提出了很多算法来解决这个问题。大体而言，可以将其分为两大类：“单社群”方法和“多社群”方法。所谓“单社群”方法，就是说网络结构中的某个节点只能隶属于某个社群，不允许出现隶属多个社群的现象。而“多社群”方法则允许用户同时隶属于多个社群。下面分别以GN算法和“最大团结构”作为这两类算法的代表对其思路进行简要介绍。

GN算法

GN算法是一种非常常用的图结构中社群自动发现

算法，最初由Girvan和Newman在2002年提出，因其有效性得到了广泛的使用。

GN算法的基本思想是：在图结构中，首先计算每条边的“介数”，然后从图中删除“介数”最大的边，如此不断循环，一直迭代删除当前“介数”最大的边，最终就形成了发现的社群。所谓边的“介数”，指的是图中任意两个节点的最短路径中经过这条边的次数。边的“介数”越大，则这条边是连接了两个或者多个社群或者圈子的多余边的概率越大，因此通过不断删除高“介数”边可以达到分离社群的目的。

GN算法比较有效，但它是一种“单社群”发现方法。就是说，图中某个节点只能属于固定的一个社群，不可能同时属于多个社群。这与实际应用场景需求有较大差异，因此成了该算法的局限。

“最大团结构”算法

“最大团结构”（max clique）是一种比较流行的能够进行“多社群”发现的算法，即图中的节点可以同时隶属于多个社群。

“最大团结构”能对图的拓扑结构进行分析，找到满足“最大团”性质的子图结构，即最大的全联通子图，每个“最大团”就是一个发现的社群。

尽管“最大团结构”算法可以发现某个节点属于多个社群，比“单社群”发现方法有更多的实用性和应用场景，但这个算法也有其局限：因为“最大团结构”要求是全联通子图，即子图中任意两个节点都有边连接，这是一种非常强的约束。真实应用中能满足如此强约束的图结构很少，这导致这个算法很多图中的节点无法归入某个社群。

HipHop算法的某些步骤中也采取了“最大团结构”的思想，但通过技术手段放松了这种约束，有效地改进了其效果。

利用HipHop算法发现社交圈

Hiphop算法利用微博用户的互动关系来自动挖掘某个用户的不同社交圈。这里的“互动”是一种总称，包括转发微博、评论微博和@其他用户等行为。如果用户A和用户B有任意上述提到的行为，则可以认为两者间有互动关系存在，且根据其频率可以赋予边不同的强度，代表了两个用户的社交亲密程度。

我们之所以使用社交关系来挖掘社交圈，是基于以下的一个基本假设：与某个微博用户进行过交互行为的人群存在于不同的小团体中，而小团体成员之内有较为密切的互动行为，不同小团体之间、成员之间交互行为较少。比如你的大学同学之间在微博上有较多互动行为，但他们和你的同事之间就很少有交互行为（如图1所示）。尽管这只是一种假设，但实际挖掘效果表明大多数情况下这种假设是成立的。

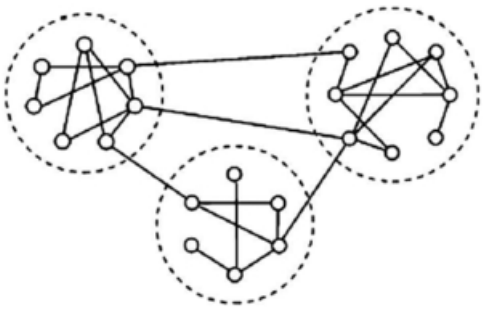


图1　不同团体的社交行为

HipHop算法的技术流程可以划分为顺序进行的三个步骤。

步骤一：从与用户有直接互动的其他用户中寻找“最大团结构”。

首先，对于微博用户A，所有与用户A在微博上有过直接互动行为的用户形成直接互动集合S。本步骤试图在集合S中找到多个“最大团结构”，即挖掘多个小团体的核心成员。

对于集合S中的节点来说，可以根据它们相互之间的互动关系构造一个图G，并在此基础上去挖掘图G中的“最大团结构”。所谓“团结构”，就是图G中包含的任意全连通子图。例如，图G中的三个节点{a,b,c}，如果它们之间任意两人都有互动关系存在，则形成了一个三节点的“团结构”。

如果某个团结构能够纳入新的节点形成新的团结构，那么它就不是最大的，最大团结构是不能容纳其他更多节点的团结构。比如上述的三节点团结构，如果存在节点d，这个节点和a、b以及c都有互动关系，那么{a,b,c,d}就形成了一个四节点的“团结构”，而如果找不到节点能够和{a,b,c}都有

互动关系，那么{a,b,c}就是一个三节点的“最大团结构”。

图的“团结构”是一个非常强的约束，因为它要求图中任意两个节点都存在互动关系。步骤一找出的某个用户A的“最大团结构”的物理含义是：与用户A有密切关系的那些用户中，有哪些是有密切联系的小团体。

步骤二：“最大团结构”在直接互动用户集合中的扩充。

步骤一找出了与用户A有过直接互动行为的集合S中形成的“最大团结构”。步骤二在此基础上，在集合S范围内对每个发现的“最大团结构”进行扩充，来发现更多属于某个“最大团结构”的其他用户。具体的扩充方式如下。

对于某个具体的“最大团结构”T，其包含若干用户，首先找到和T中用户有过互动行为，同时又在集合S中的其他用户，我们简称这个集合为U。对于U中的某个用户w，我们需要判断是否应该将其扩充进入“最大团结构”T。目前的判断标准采取如下公式：

$$Utility(G)=\frac{\sum_{e\in E_{in}} Weight(E_a)}{\sum_{e\in E_{out}} Weight(E_b)}$$

假设G是最大团T将用户w融合后形成的新图，公式的分子部分代表新图G中所有节点内部边的权重之和，而分母部分代表图G中所有节点和图G之外的任意节点形成的所有边权重之和。如果Utility(G)函数比未扩充节点w的原图结构T的效用函数Utility(T)值大，那么认为将节点w扩充进入T是合理的，否则不能将节点w扩充进入图T中。有这个函数作为标准，我们就知道集合U中的用户哪些该扩充进入团结构T中，哪些该被舍弃。

之所以采取上述公式作为判断标准，是基于之前提到的如下假设：一个社交圈子成员之间互动关系密切，而圈子成员与圈子外成员之间的互动关系不是很密切。上述公式就是这个基本假设的具体体现，分子部分是衡量圈子成员内部的关系紧密程度，而分母衡量的是圈子成员和圈子外成员的关系紧密程度。

从公式可以看出，如果圈子成员之间互动越多，而与圈子外成员互动越少，则效用函数越大，也就是说这个圈子越紧密。

如果对于集合U中所有后续扩充用户都采用上述公式进行判断取舍，来做出是否将这个用户扩充进入“最大团结构”T的决策，那么就完成了T的一轮扩充，形成了扩充后的新集合T'。对于T'来说，仍然可以采取上述扩充方法不断外扩。“最大团结构”T外扩的终止条件是：如果对于集合U中所有用户，做出的决策都是不进行扩充的，那么此时已经达到了扩充的边界，可以停止外扩，形成最终扩充结果。

如果对步骤一中发现的所有“最大团结构”都采取上述方式外扩，就完成了步骤二的任务。可从上述过程看出，步骤二是对步骤一的扩充阶段。

步骤三：与用户有“二级互动”关系的其他用户集合中的扩充。

所谓用户A的“二级互动”用户集合，是指与用户A有直接互动的用户形成集合S，而与集合S中任意一个用户有互动行为的所有其他用户形成了二级互动集合。

从步骤二的结果来看，步骤二完成了对“最大团结构”的扩充，在直接互动用户集合中找到了不同的社交圈子。步骤三首先将直接互动用户集合S扩充为二级互动用户集合，然后采取与步骤二类似的方法继续向外扩充，这样就形成了HipHop算法的最终结果，形成了用户A的多个不同社交圈，而任意一个其他用户B可能同时属于用户A的多个社交圈。

通过上述三个步骤，就可以利用微博互动关系自动挖掘出某个用户的社交关系圈。对于微博的海量用户而言，只要对每个用户都依次采取上述步骤，即可获得最终结果。这可以采取大规模并行计算来快速实现。

下面结合一个例子讲述HipHop算法。以“李开复”作为示例，说明上述步骤及其中间输出结果。

对于步骤一，首先找到与“李开复”有过互动的微博成员形成集合S，之后在集合S里采用发现“最大团结构”的方法，可以得到最初的5个“最大团结构”：

- 最大团1（创新工场有关）：王肇辉/蔡学镛/周源/张亮/徐磊Ryan
- 最大团2（互联网媒体相关）：keso已被XX/牛立雄/金磊
- 最大团3（财经投资相关）：徐小平/爱国者冯军/潘石屹/杨澜
- 最大团4（创新工场有关）：郎春辉/罗川/袁聪iw/应用汇
- 最大团5（企业家相关）：曹国伟/江南春/吴征bruno/蒋锡培

经过步骤二，对原始的5个最大团在集合S中进行扩充，每个原始的最大团都有不同程度地扩大，其新扩充进的成员范围在3~10个。

步骤三，首先将直接互动成员集合S扩充为二级互动成员集合，即将与集合S中成员有过互动行为的微博用户形成新的更大范围的集合。根据前面讲述的扩充方式，5个最初的“最大团结构”获得进一步扩充，最后形成了多个不同的社交圈。

经过人工评估，HipHop算法挖掘出的社交圈有较强的社交内聚度，同时也满足算法设计之初设定的几个约束条件，因此具有很强的实用性。同时，经过大量实例分析，我们发现在微博中形成的社交关系和IM形成的社交关系有较大的差异，大部分用户的微博中的社交关系以同事关系和兴趣关系为主，而IM中形成的社交关系则以亲友、同事、同学等线下关系为主，这可能反映了社会化媒体和传统社交网络的区别所在。



张俊林

《这就是搜索引擎：核心技术详解》作者，新浪微博研发人员，中科院软件所博士，主要研究方向为自然语言处理、搜索技术、推荐系统及机器学习。

责任编辑：杨爽（yangshuang@csdn.net）

pongo庞果

寻找机遇 创造未来

庞果职位全新推荐

■详细信息请参见pongo网站：www.pongo.cn

北京世联互动网络有限公司成都分公司

NHN 拥有世界排名前列的搜索门户；是全球知名的网游门户。NHN 成都研发中心专注于互联网核心技术的开发及创新，为您提供高起点的事业平台及完善的福利待遇。NHN 将结合挑战、激情、创新、变革的运营理念，努力为中国市场营造全新的网络世界。Join Us！

现诚聘如下职位：

- 数据库引擎研发工程师
- 高级开发工程师(C/C++方向)
- Windows 应用开发工程师
- Web 前端研发工程师
- 高级开发工程师（Java 方向）
- QA 工程师

简历投递：http://org.pongo.cn/Org/Details?ID=346406

地址：四川省成都市锦江区三色路 38 号博瑞创意成都大厦

北京呈天时空信息技术有限公司

T4GAME 呈天游是一家中国领先的多平台在线游戏开发商和运营商。公司的使命是为用户创造优秀的随身的娱乐体验！

现诚聘如下职位：

- C 引擎工程师
- Android 客户端
- Flash 业务逻辑工程师
- 3D 引擎工程师
- Java 服务器逻辑工程师
- Java 开发工程师
- 运营总监
- C 业务逻辑工程师

简历投递：http://org.pongo.cn/Org/Details?ID=346366

地址：北京市朝阳区八里庄东里 1 号莱锦文化创意园 CN15

新华网股份有限公司

新华网是由新华社主办的大型网络文化企业，现因事业快速发展诚聘英才。

www.news.cn 新华网 NEWS www.xinhuanet.com

现诚聘如下职位：

- Java 高级开发工程师
- Android 开发工程师
- 网站系统架构师
- 数据挖掘开发工程师
- iPhone 开发工程师
- Web 前端开发工程师
- 数据仓库开发工程师
- 搜索技术开发工程师

简历投递：http://org.pongo.cn/Org/Details?ID=346307

网址：www.news.cn　www.xinhuanet.com

Hash函数与带有防欺骗的承诺

文 / 顾森

互联网给我们的日常生活带来便捷的同时，也带来了信息安全隐患。很多安全问题可以简单地利用物理手段解决，但在纯信息交流层面，这可能会困难许多。在上期杂志中，我们看到了“秘密共享的门限方案”及其应用。本期让我们来研究一下带有防欺骗的承诺。

承诺的防欺骗性

假如你碰到了一个宣称可以预知未来的人，他说他知道下周的彩票中奖号码。你肯定不会相信，便用激将法让他说出来。不过，他却一本正经地说：“不行，我虽然能预测未来，却不能把它说出来，否则会产生蝴蝶效应，改变这个宇宙既定的将来，导致危险的时空悖论。”

这个“先知”真是天才呀！能预言未来却不能说出来，这样就永远不能证伪了。

不过，治他的方法也不是没有。比方说，可以叫他把预测结果写在一张纸上，锁进一个盒子里。然后，你拿走盒子，他拿走钥匙。彩票中奖号码公布后，你们再碰个头，把盒子打开，看看当初的预测结果是否正确。这样便能让他做出一个谁都不能看见，但他今后也不能抵赖的预测。我们把这样的协议叫做“带有防欺骗的承诺”。

只可惜，这种方法有一个局限：它只能在现实生活中使用。如果你在网上遇到了自称能预知未来的人，如何让他做出防欺骗的承诺呢？

Hash函数解决方法

可能有人会说，为什么不模拟刚才的物理手段，让他给预言加一个密呢？比方说，让他给下周的中奖号码上加一个很大的整数，然后把结果告诉你；这个很大的整数就是解开中奖号码的密钥，由他自己保管，并在今后适当的时候交出。仔细想想你会发现，这个方案显然不行，因为到了验证预言时，他可以谎报这个大整数，让密码解开后是任何一个他想要的数。为了防止他耍赖，能否让他事先就把密钥公布出来呢？这也不行——知道了密钥后，你就能直接获得中奖号码原文了，这样便失去了保密的意义。

我们注意到，传统的加密方法不能公开的原因就是，知道了加密方法也就知道了解密方法，只需把加密方法反过来做就行了。有没有一种加密方法，使得即使你知道了加密的方式，也不能恢复出原文呢？有的。只需要在加密过程中加入一些不可逆的数学运算就行了。比方说，可以约定这样一种加密方法：先取中奖号码的正弦值的小数点后八位数字，得到一个八位整数；再求中奖号码与圆周率前六位数字形成的整数（314,159）之和，取该和的平方的第3位到第10位，又得到一个八位数；最后计算这两个八位数的和除以456,789的余数。假如他预言的中奖号码是1,234,567，那么对1,234,567进行上面这一串操作后，将会得到244,685。但即使知道加密的过程，你也不能把244,685还原成1,234,567。事实上，1,234,567甚至

不一定是唯一解，很可能有别的数加密后也会变成244,685。上述加密方法能把任何数都加密成一个小于456,789的非负整数，因此必然会出现不同的数加密成同一结果的情况。这就意味着，这种加密方法是会丢掉原始信息的。我们不妨把这种不可逆的加密方法叫做“单向加密”，加密所用的函数则通常被称为“Hash函数”。

在密码学中，MD5是一种常用的Hash函数，它可以对任意长度的原始数据进行一番折腾，最后将其转换为一个32位的十六进制数。为了能够处理任意长度的数据，并产生足够混乱的输出结果，MD5算法利用了很多Hash函数都在使用的Merkle-Damgård结构：把数据每512个bit分成一组，然后对第一组数据进行MD5指定的变换，将结果加入第二组数据参与同样的运算，所得结果再加进第三组数据，以此类推，不断迭代直到得出最终结果。若剩下的一组不足512个bit，则在整个数据末尾添加一个“1”，然后添“0”补足。为了进一步加强算法，当添“0”添到还剩64个bit时，必须再填进一个等于原始数据总长的64位整数。这样一来，数据任何地方的任何微不足道的差异，都会影响到整个输出结果。例如，本文第一段的MD5值（从“假如”到最后的反引号）是ED57471E12F8478172A74AB8C3D8B39E，如果在其最后多加一个空格，MD5值就变成了BCE3B601D14FC874D6C22DF2F1666C43。

由于其单向性，这些Hash函数不能用于普通的信息加密传输，却有很多其他应用，做出带有防欺骗的承诺便是一例。

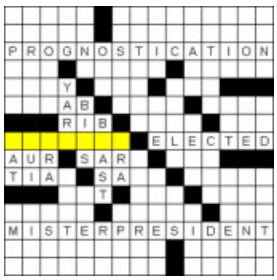
在前面的例子中，拿到244,685这个数后，你完全无法推出他究竟做了什么预测；到了验证预言时，只需要让对方宣布当初他的预测1,234,567，你来检验一下对1,234,567进行运算后是否会得到244,685就行了。

Hash函数的冲突

不过，这个方法有一个局限：如果他宣称能预测某只股票会涨还是会跌，上述方法就有漏洞了。比方说，你们可以约定，数字1表示股票会涨，数字2表示股票会跌，然后让他用刚才的那套方法把预测结果加密发过来。如果他告诉你的结果是316,554，那你只需要分别试一下1和2加密后分别得多少，就知道原始数据是1还是2了。原始数据的取值太有限，让穷举式的“暴力破解”变得易如反掌。怎么办呢？可以想办法硬把原始数据的取值范围扩大。比如，约定所有个位数字为1的数都表示股票会涨，约定所有个位数字为2的数都表示股票会跌。假如对方预测股票会涨，他可以选取任意一个末位为1的数，对其进行加密，这下你便没办法暴力破解了。不过，这样一来，对方又有赖皮的机会了：他可以精心构造出“两可解”的情况。刚才我们说了，Hash函数可能会把不同的原始信息加密成同一个结果，因此完全有可能出现这样两个数，它们的末位分别是1和2，但加密后的结果相同。对方可以给出这个加密结果，并在协议后期宣布对自己有利的那个原始数。

虽然找到这样的例子可能会很难。不过，人才总是有的，再不可能的事也有人办到。1996年11月5日，美国总统大选日，竞争主要发生在民主党人克林顿与共和党人鲍勃·多尔之间。这天早晨，《纽约时报》像往常一样刊登了一个纵横字谜，奇怪的是，正中间那一行是两个长度均为7的单词，根据提示这两个单词连在一块儿就是“明天报纸的头条新闻”。可以推出，后一个单词是ELECTED（赢选），但是前一个单词是什么？当日晚上，大选结果出炉，克林顿以379对159的优势击败对手获得连任。第二天早晨，《纽约时报》也公布了昨日谜题的答案，中间一行正是CLINTON ELECTED（克林顿赢选）。

纽约时报是怎么预测到大选结果的？其实秘密很简单——这个纵横字谜有两个解，在CLINTON的位置填写BOBDOLE（鲍勃·多尔）也是完全满足要求的。这看上去很不可思议，因为每一个字母的改变都会连带着影响到该列的单词，甚至产生连锁反应，推翻整个局面。然而，天才的纵横字谜作者Jeremiah Farrel就做到了——他在相关的地方巧妙地设计了一系列非常漂亮的单词提



示。例如，总统名字的第一个字母所在列的单词提示是“黑色的万圣节动物”，答案既可是CAT（猫），又可是BAT（蝙蝠），填C和B都是可以的。

这个故事的程序员版在2007年11月30日重新上演。几个密码学家在一个网站上宣布，他们利用一台PlayStation 3游戏机成功地预测了2008年美国大选结果，并给出了一个带有防欺骗的承诺：他们写好了一个说明预测结果的文档，文档的MD5值为3D515DEAD7AA16560ABA3E9DF05CBC80。但事实是，这帮人已编好了12份PDF文档，分别描述了12种不同的预测结果，但它们的MD5值都是刚才那串十六进制数！你可以在这里下载到这12份奇特的文档：http://www.win.tue.nl/hashclash/Nostradamus。

精心构造出MD5函数值的冲突似乎是极其不可能的事。然而2005年，我国的王小云等人演示了一种实用的MD5冲突构造法，MD5从此变得不再安全。

找到这样的“Hash冲突”显然更加困难，尤其是考虑到这里使用的是更庞大、更专业的Hash函数。MD5的输出结果是一个32位的十六进制数，这大概有340万亿亿亿亿种可能性；并且，MD5的函数运算过程足够混乱，原始数据的任何一处细微变化都会被放大。因此，精心构造出MD5函数值的冲突似乎是极其不可能的事。然而2005年，我国的王小云等人演示了一种实用的MD5冲突构造法，MD5从此变得不再安全。随后，不断有人提出了构造MD5冲突的算法，使得在普通笔记本电脑上构造MD5冲突的时间从数小时减少到了数秒。不过，如此找到的冲突数据，往往都是一些看上去毫无意义的随机串。更具实用价值的是，2007年，Marc Stevens、Arjen Lenstra和Benne de Weger发现了一种可以任意指定前缀的冲突构造方法。也就是说，任意给定两段数据P_1和P_2，利用他们的方法可以在一到两天的时间里内找到两个后缀S_1和S_2，使得P_1+S_1和P_2+S_2的MD5值是相同的（这里“+”表示字符串连接）。于是这帮人准备了12份内容不同的PDF文件作为指定前缀，并在每个文件的末尾都预留了一个隐藏的bitmap图片元素，然后两两构造冲突，所得后缀正好填进隐藏的bitmap元素里。这样一来，他们便得到了6对MD5值相同的文件，P_1+S_1和P_2+S_2，P_3+S_3和P_4+S_4，一直到$P_{11}+S_{11}$和$P_{12}+S_{12}$。他们构造冲突时有意加了一条限制：所有的P_i+S_i长度都必须相同，并且都是512个bit的整数倍。

MD5算法有一个非常隐蔽的缺陷：假如有两段MD5值相同的数据，如果这两段数据的长度相等，并且都是512个bit的整数倍（即数据正好成组），那么加上任意一串相同的后缀后，两段数据的MD5值仍然相同。这都得怪Merkle-Damgård的按组依次迭代结构：若某次迭代后的结果相同，正好此后的数据也完全一致，那么后面的一切都唯一确定了。我们便可以继续套用冲突构造算法寻找S_5和S_6，使得$P_1+S_1+S_5$、$P_3+S_3+S_6$的MD5值相同，这样$P_2+S_2+S_5$和$P_4+S_4+S_6$也都自动地拥有了同样的MD5值。同时，构造冲突时需要保证所得数据的长度仍然相同并等于512的整倍数，从而让MD5值已经相同的数据今后不会被破坏掉，以便让这个小伎俩可以继续使用下去。12重冲突全部构造完毕后，再在每个文档末尾添加相同的文件尾，最后便能得到12份合法的PDF文档。它们的MD5值全都相同，但所含内容截然不同。

因此，作为Hash函数，MD5已不再安全了。一个安全的Hash函数必须是“抗冲突”的。从这个角度来讲，目前常用的SHA-1则更加安全一些。另外，我们可以在每次实施协议时都改变一下协议的细节，例如每次都临时为“预言承诺者”指定一个新的Hash函数，或者每次都要求承诺内容的原文必须满足某个古怪而苛刻的要求，从而破坏掉对方精心准备的两可解。这样一来，整个协议就完美了。

在生活中的应用

其实，整个协议并不只在揭穿超能力者时才有用。我们生活当中有很多地方都可以用到带有防欺骗的承诺。有一次，我在战网上和别人打星际，打出了一个非常搞笑的局面：两边的兵都一个不剩，两边的钱也都不够造东西了，双方都完全丧

失了战斗能力。但双方都还剩有建筑，因此都不算输。此时，必须有一个玩家主动认输，先退出游戏，才能结束僵局。该谁先退呢？我和他便在游戏中互发消息谈论了起来。其实，在现实生活中，这很好解决：来玩一次石头剪子布就可以了。但是，怎么在网上玩石头剪子布呢？总不能让一个人先发消息说“我出的是剪子”，另一个人回复“哈哈，我出的石头”吧。这时，就要用到带有防欺骗的承诺了。我们可以利用前面讲的方法，各自向对方承诺自己要出什么拳，然后双方再公布自己出的拳，让对方验证自己并没撒谎。更简单的方法就是，我在1和2之间想一个数，然后把我想的数加密告诉对方，由对方来猜我想的数是多少，猜对了我就认输退出，猜错了他就认输退出。对方做出猜测后，我再公布加密前的原始信息，以证明我没有耍赖。

我们常在电视上看到这样一幕：一位老太太兴冲冲地走上台去，翻过3个商标牌，发现上面尽是5块钱、10块钱的小奖，垂头丧气地回到观众席；然后主持人会跑出来，边翻着另外几个牌子边说，1000元的大奖在这个后面，800元的在这里之类的。为什么主持人要演出“事后揭大奖”这一幕呢？道理很简单，节目组想通过这个“验证过程”告诉观众，这个环节不是骗人的，大奖真的就在这里面，只是刚才那家伙运气背了没摸到而已。摸奖前宣称有大奖，摸完奖之后还能证实大奖真的存在，这也是带有防欺骗的承诺。

但是，我们喝饮料参与开盖有奖活动时，就会有被欺骗的感觉：你说中奖率是千分之一，我凭什么相信你呢？那么，有没有办法让开盖有奖活动的中奖率变得透明呢？有的，我就想过这么一种方法：开盖后你将得到一个参与活动的序列号，把这个序列号短信发送给活动举办方参与抽奖。此时，活动举办方的服务器从1到1000中随机生成一个整数，并向你做出带有防欺骗的承诺。你需要猜出服务器生成的数是什么，如果猜对就能中奖，如果猜错就结束游戏。发送了你的猜测结果后，服务器将发来Hash前的信息，证明自己没有撒谎。

在网络中，Hash函数还有很多其他的用途。首先，Hash函数可以当做校验和（Checksum）使用。如果有人传给了你一段数据，怎么才能知道你收到的是否和原始文件一模一样呢？万一有个别字节传错了怎么办？一个简单的方法是再传一遍比较一下，然而这种方法耗费成本太大。更聪明的方法则是，用某种公开的、标准的、确定性的算法生成文档的“摘要”，或者“指纹”。文档中任何细微的差异都会让“摘要值”或“指纹值”改头换面。双方只需要算出各自手中的文件所对应的值，比一比这个值便能确定是否是同一份文件了。大多数Hash函数都能承担这样的任务。在网上发布一个应用程序时，作者往往会公布文件的MD5值或SHA-1值，方便大家下载后检验文件的完整性和正确性，避免攻击者别有用心地替换掉原始文件。

在服务器中储存用户密码时，我们往往会避免直接储存密码明文，而是储存密码的Hash值。用户登录时提供的密码只要Hash之后与服务器所存的相符，即可通过身份验证。这样做的好处就是，即使攻击者获得了数据库上储存的Hash值，也无法反推出满足要求的密码。不过，对于常用的密码，网络上有很多预先算出的“常见密码MD5值表”可供反查（俗称“彩虹表”）。另外一个小缺陷则是，如果两个密码的Hash值相同，我们（基本上）可以得出这两个账户的密码也是相同的，虽然不知道具体密码是什么，但这一信息也足以产生危害。所有这些缺陷都可以用一种俗称“加盐”的方法来解决，即在用户设定密码时，服务器自动为用户密码添加一个随机前缀，然后把“前缀＋密码”的Hash结果与前缀本身都储存起来。如果用户登录时提供的密码加上他的随机前缀后，Hash值正好能对上，即通过身份验证。这种方法将会有效地防御字典式的攻击。



顾森

网名Matrix67，数学爱好者。2005年开办数学博客http://www.matrix67.com，至今已积累上千篇文章，有上万人订阅。新书《思考的乐趣》已在图灵公司出版。

责任编辑：卢鹃翔（ludx@csdn.net）

Node.js的核心与红利

文 / 田永强

唯有明晰历史，才能了然当下，预知未来。作者从历史角度解读Node.js，帮助读者透过猜忌和谣言，看清真实的Node.js，了解Node.js的核心与红利。

令人惴惴不安的Node.js

我们越来越频繁地看到关于JavaScript的新闻，刚开始谈到的是引擎性能提升，后来则是由HTML5和Node.js带来令人叹为观止的应用。如果只看表面，容易让人产生以下各种误解。

- Node.js的出现是让前端工程师羞辱后端吗?
- Node.js肯定是几个前端工程师在实验室里捣鼓出来的。
- 为了后端而后端，有意思吗?
- 异步是反人类的。
- 怎么又发明了一门新语言?
- JavaScript承担的责任太重了。
- 直觉上JavaScript不应该运行在后端。

JavaScript的诞生

JavaScript 1995年，诞生于网景公司，因为当时需要一种脚本语言协助Navigator做简单的操作。Brendan Eich受命操刀实现它。当年网景公司与Sun合作紧密，不懂技术的高管希望得到一个类似Java的脚本语言，期望它能像Java一样风靡。而Brendan Eich原本期望进入网景公司做Scheme研究，却接到了一个他不喜欢的任务。形势所逼，他在10天的时间里完成这门语言的设计。它唯一符合高管们期望的地方就是它长得确实像Java。

这看起来这本是一件糟糕的事，所幸Brendan Eich的才华确实令人高山仰止。高管期望的Java+Script实际上是C+Scheme+Self+Java的产物。它本不该叫JavaScript，却获得了这个名字，谈起这件事，许多开发者会拿“雷锋”与“雷峰塔”的关系开玩笑。

JavaSript诞生后另一件糟糕的事出现在第一次浏览器大战时。微软在IE中实现了JScript，与JavaScript十分相似。网景为了保护自己，过早推动了这门语言标准化的进程，在1997年ECMAScript262规范颁布。

两件事情叠加在一起，使得JavaScript还没有足够的经历就被过早地出生和定型。它就像一个早产儿，之后的成长中，处处透露出不成熟的气息。所幸福祸相依，看起来糟糕的事情背后，依然留下了两个拐点，为JavaScript保留了优秀的基因。

- 引入了Scheme的函数式，函数成为一等公民。函数式编程里的高阶函数、偏函数都充满了灵活性，它们在JavaScript中得以发挥。
- 引入了Self的原型链。得要感谢Brendan Eich的这个决定，不然现在大多数人可能只知道Java的模版继承，却不知道另一种继承方式叫做“原型链继承”。

最早的服务端JavaScript

很多人觉得JavaScript不适合运行在服务端，但这

表1 JavaScript和其他语言性能对比

语言	时间 (user)	名次	版本
C with -O2	0m0.187s	#0	i686-apple-darwin11-llvm-gcc-4.2 (GCC) 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)
Node + CPP module	0m1.417s	#1	v0.8.8, gcc -O2
Go	0m1.664s	#2	Go version go1.0.2
Luajit	0m2.463s	#3	LuaJIT 2.0.0-beta10 -- Copyright (C) 2005-2012 Mike Pall.
Node.js	0m2.826s	#4	v0.8.8
PyPy	0m29.650s	#5	Python 2.7.2 (341e1e3821ff, Jun 07 2012, 15:42:54) [PyPy 1.9.0 with GCC 4.2.1]
Lua	0m42.944s	#6	Lua 5.1.4 Copyright (C) 1994-2008 Lua.org, PUC-Rio
Python	1m19.029s	#7	Python 2.7.2
PHP	1m43.355s	#8	PHP 5.3.13 with Suhosin-Patch (cli) (built: Jun 20 2012 17:05:20)
Perl	2m30.444s	#9	This is Perl 5, version 12, subversion 4 (v5.12.4) built for darwin-thread-multi-2level
Ruby	3m32.074s	#10	Ruby 1.8.7 (2012-02-08 patchlevel 358) [universal-darwin12.0]

是错的，尽管一直处在Java的阴影下，但网景还是期望JavaScript能够既运行在浏览器中，也能运行在服务器端。网景最早的服务端JavaScript项目叫做LiveWire，在今天的互联网上仍能看到它的遗迹（http://www.datacraft.com/livewire.html）。当JavaScript运行在服务端时，PHP还没有盛行（PHP最早版本于1995年发布）。

服务端JavaScript之殇

但理想与现实之间总是存在差距。浏览器市场份额之争使得JavaScript在浏览器中固化且无可替代。另一厢，Java风行互联网，服务端JavaScript完全没有抢眼球的机会。并且随着网景在浏览器战争的失利，后端JavaScript从此式微。

服务端JavaScript遗民

尽管最初这条道路十分崎岖，甚至荒芜，但行人总是有的。因为这是一门图灵完备的语言。Rhino作为Java实现的JavaScript引擎一直存在。前端JavaScript也一直作为表单校验功能为Web 1.0发光发热。这一切都让人觉得JavaScript只是一门辅助语言。这也是造成部分人误解它的原因，认为JavaScript的能力仅限于此，在看似职责分明的架构体系中，扮演着基本角色。后端JavaScript尽管存在，但极少有应用场景。

第二次浏览器大战

直到2006年，JavaScript还处在打酱油的位置。但Web 2.0和第二次浏览器大战的到来，催生了翻天覆地的变化。Web开始趋向应用化，JavaScript的角色一下子变得重要起来，从小脚本开始出现大规模编程需求。

另一个变化则是JavaScript引擎在V8的引领下，速度大幅提升。新闻中时常出现“提升x倍”这样的词。这带来了一个以往难以想象的事实——在脚本语言中，JavaScript的速度已在顶尖之列。我的同事苏千提供了斐波那契数列的测试结果，这是一个纯考验CPU的运算：

```
function fibonacci(n) {
    if (n < 2) {
        return n;
    }
    return fibonacci(n - 2) + fibonacci(n - 1);
}

console.log(fibonacci(40));
```

为了让比较更为直观，加入了C和Go两个静态语言作为参考（如表1所示）。

可以看到，JavaScript在V8平台的表现与Luajit相差无几。借助C++模块，性能可以更进一步提升。如果需要了解最新的对比结果，可以访问这个页面 http://fengmk2.github.com/blog/2011/fibonacci/nodejs-python-php-ruby-lua.html。

2006–2009年，JavaScript得到了进一步发展，效果主要体现在两个方面：

- JavaScript引擎性能继续大幅提升；
- 趋向应用化，出现大规模编程的需求。

由于ECMAScript过早标准化，时至今日，JavaScript标准的核心其实只是其完整语言所涉及的极小部分，与DOM相关的API部分则被归于W3C规范。

CommonJS

沉寂许久之后，服务端JavaScript开始有所复苏。但那时JavaScript的表现还严重受到浏览器的制约，后端JavaScript也并没有标准化，呈现出一种混乱分离的状态。这时CommonJS出现了。

CommonJS基于语言现状和一直想要充分发挥JavaScript能力的愿望，开始制定JavaScript超出ECMAScript262、浏览器DOM与BOM API之外的规范。这些规范包括模块、包、二进制、流、Buffer、网络等。

这是一个理想主义的做法，但直到Node.js出现，很少听闻它的消息。原因还在于人们的成见，认为JavaScript除了运行在浏览器中，没有其他作为。它致力于将JavaScript作为命令行工具、桌面应用、服务甚至生态系统的理想，尽管在少部分人的帮助下得以实现，但在推广中收效甚微。

理想很美好，现实很骨感，但这还是埋下了又一伏笔，对于未来的发展至关重要：

CommonJS规范的出现可以指导JavaScript实现大规模编程。

这套规范后来不仅影响到了服务端JavaScript，也影响到前端JavaScript。

Node.js的逆袭

Node.js的出现终于让开发者开始关注服务端JavaScript，不过Node.js的作者并不是前端工程师，按时髦话来说，Ryan Dahl是一枚资深C/C++码农。

他没有帮助前端工程师羞辱后端的愿望，也没有这份耐心。选择JavaScript，属于偶然中的必然。Ryan Dahl最初利用Ruby写了一个叫做Ebb的Web服务器。像其他资深C/C++码农一样，他将心思花费在服务器的实现上，例如放弃原来的Apache fork/prefork、多线程等模式，因为大家都意识到单线程异步的方式比原来的多线程或多进程具备更高的性能，Nginx就是这样的产物。此外，Ryan Dahl还意识到，尽管面向客户端连接可以通过异步来提升性能，但在业务层面的一个同步I/O常让他在提升性能上所做的努力徒劳无功，因此他要的是面向客户端的异步，面向后端业务，也是异步的一个服务器。

他开始评估几门语言实现这个Web服务器的可能——Ruby因为VM的性能原因被舍弃了；尽管Ryan Dahl擅长C/C++，但他深知开发C/C++的门槛十分高。排除几种方案后，Lua是最接近他期望的语言。但Lua已有的I/O库是同步的，他知道，如果在Lua中推行异步I/O，将会因为不符合人们的习惯而被抛弃。

这时JavaScript出现在了他的面前。也许是历史的必然（或巧合），前文描述的伏笔或多或少影响了Ryan Dahl。JavaScript的单线程、事件驱动的方式也十分贴合Ryan Dahl的理念。更重要的是，异步在JavaScript中已广泛使用（例如Ajax），服务端JavaScript的停滞给Ryan Dahl留出了巨大空白——于是Node.js诞生了。

Node.js的出现给服务端JavaScript带来了新气象，CommonJS也应用到了Node.js的模块和包上。凭借V8和异步I/O带来的性能优势，一扫曾经的阴霾。尤其是将异步引入到业务层，非阻塞特性使得JavaScript的语言模型显得足够独特。

Node.js与Nginx之间也可以稍作比较。通过我们的线下测试，发现Nginx的性能比Node.js的HTTP模块要好很多。但Nginx考量的是面向客户端，后端业务方面依然是受具体业务影响。而Node.js则可以利用异步I/O来实现业务并行，以提升效率。从这点看，Nginx没有Node.js灵活。此外，Node.js后来的发展方向不再单独是一个Web服务器，而是一个面向网络的平台，它甚至可以是TCP服务器，或者变身为远端服务器的客户端。

异步的利与弊

异步带来的好处主要是性能方面，如果没有异步，就没有高性能服务器。异步让CPU与I/O之间重叠利用，可以很好地利用硬件，提高吞吐率。异步早就应用在操作系统层面中，在操作系统内部，处处是信号量，只是在业务层面出现较少。将Node.js广泛应用异步到业务层，在面向网络时的效果是明显的。

为了直观地展示在网络中应用异步I/O的重要性，下面这组数据必不可少：

表2 网络中的I/O开销

I/O开销	花费CPU时钟周期
CPU Cache L1	3
CPU Cache L2	14
RAM	250
Disk	41000000
Net	240000000

我给出的题目很简单：如果在分布式环境中，需要从不同的地方调用10次不同的数据，时间消耗为t_1到t_{10}不等，总计消耗时间该为多少？如果没有异步I/O，同步方式将会花费很多时间，而理想状况下的异步调用则为$max(t_1,...,t_{10})$。

异步令人不满的地方在于流程控制会带来麻烦。但以日常生活为例，生活中处处是异步：排队到窗口买饭是同步，坐在桌子上点菜则是异步；网购是异步，自己跑去超市买东西拎回家则是同步。我们可以发现生活中的异步并未给我们带来多大烦恼，相反，同步的痛楚可能更明显。但在代码中，却又是相反的。不过好在问题总是可解的：将逻辑分发，或者采用赵劼开发的Wind.js都是可选方案。

拥抱开源与生态圈成型

随着Node.js带来服务端JavaScript的进步，如今JavaScript的生态圈已经形成。这里有个有趣的现象，CommonJS立下的志愿最初没有被实现，但随着Node.js采用CommonJS的部分规范，Node.js具备了理论上的指导。模块、包规范的应用，使得Node.js不像最初浏览器中的JavaScript一样无序混乱。Node.js在应用过程中催生了NPM工具，而NPM工具的广泛应用则起到了推广CommonJS规范的效果。这是一个种因得果的循环，有理想，则最终有其实现。

另一件类似的现象体现在V8与Node.js的关系上，Node.js因为V8的优异性能，选择以它作为基础构建平台，但随着Node.js不同于浏览器的应用场景，又反过来促进V8的性能提升。

Node.js自身开源，同时也推动开源，NPM上的模块数量如今已超过14000，日下载量接近50万次，社区的活跃程度史无前例。尽管我们知道Node.js继承了一些JavaScript带来的缺点，但在如此强大的社区面前，在一个解决方案比问题还多的社区面前，想没有信心都难。

作为一家企业，我们在淘宝内部也搭建了NPM的镜像服务器，充分利用NPM来管理内部私有模块，减少拷贝粘贴行为，将一切可以模块化的功能或方案尽皆做成包，共享给兄弟项目或者发布到官方NPM上，回馈给开源社区。

如果一个开源社区的循环是良性的，我们没有理由不信任它，因为成型生态圈，使它具有持久的生命力。但有一件事必须明了，尽管这个生态圈已然成型，但Node.js自身与现有技术并不是非此即彼的，发挥异步并行的优势，与其他系统异构共存，将会是很长一段时间内的主题。即便是理念最为相近的Nginx，也是可以与Node.js互相协助的。

总结

也许是早产注定了JavaScript留给开发者发挥的余地够多，尽管不是有意缺少，但确实体现出了Less is More的精髓。按赵劼的话说，尽管JavaScript有很多毛病，但它设计得还不赖。相信后续的ECMAScript标准化能够修缮过去的问题。在经历了多年的发展以后，JavaScript已不是从前的它，从核心上衍生出的红利也是显而易见的，生态圈及前后端语言打通，使得更多开发者可以参与其中。随着Node.js在服务端大展拳脚，第二波JavaScript浪潮已经到来，如果依旧怀抱过去的观点，等着被恭喜Out吧。



田永强

花名朴灵，2011年在上海组织CNode社区的线下分享会。2012年加入淘宝的数据产品团队专职参与Node. js开发。同时他还在InfoQ主持《深入浅出Node. js》专栏。

责任编辑：卢鹄翔（ludx@csdn.net）

前端模块加载策略

文 / 蒋定宇

随着前端开发技术的蓬勃发展，需求大量移往前端处理、甚至将网页做成类似Gmail、Google Calendar等交互型应用程序。这个趋势使得CSS与JavaScript的代码量大幅增加，也使得加载许多现成的“库”来满足复杂的需求。本文将结合工具及范例，介绍加载JavaScript和CSS的数种方法与模式，进而引出载入前端模块的理想方式。

模块化开发的兴起与疑问

过去开发JavaScript并没有很好的模块架构管理依赖关系与加载机制，当项目变复杂，往往难以维护。直到Node.js采用CommonJS 1.0的模块规范流行后，大家才开始热烈讨论浏览器端的模块架构规范与实现。现在AMD规范与RequireJS实现可说是整个前端界的当红炸子鸡，为前端程序员提供了一个开发JavaScript模块时定义依赖关系和加载的标准。

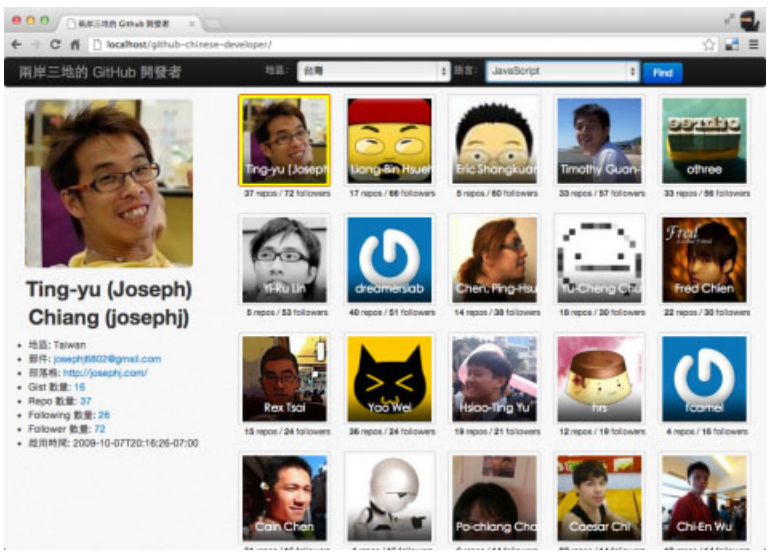


图1　范例网页：两岸三地的开发者

但对于网站的前端开发来说，我们要考量的并不只是动态加载的JavaScript文件，还得考量CSS如何载入。甚至当我们采用了类似雅虎把页面切割更细的“页面模块开发模式”后，又该怎么做呢？

模块定义

模块的定义有千百种，大多是指可以重复利用的元件，让开发者可以直接采用，避免重新打造轮子。本文所指的模块则是任何可以被载入的HTML、CSS、JavaScript文件。例如jQuery在载入后可以简化许多JavaScript操作，而Twitter Bootstrap则让不善美工的程序员轻松拥有亮丽的版面。两个模块一挂入网页，真可说是神兵利器，可以满足前端程序员许多的需求。

页面模块开发模式

前面提到雅虎的“页面模块开发模式”，则是将页面依结构切分成一个个小的 模块，原则上每个页面模块都有各自的HTML（例如PHP是用include_once来载入）、CSS与JavaScript文件。好处是让每个开发人员都能专注于单一模块的开发，而非去烦恼整个页面的架构与布局。



图2　两岸三地的开发者网页的页面模块结构图

重复利用在这样的开发模式下并不是最重要的事情。

在两岸三地开发者的网页（如图1所示），其结构（如图2所示）共有三个页面模块：上面的“条件筛选模块”（_filter）、左侧的“资讯模块”（_info）、右侧的“列表模块”（_list），都有各自的HTML、CSS与JavaScript。

这样的模式是目前业界的开发主流，后面的介绍将以这个开发模式为思路基础。

传统加载模式

传统加载模式就是使用<link/>载入CSS、<script/>载入JavaScript。当我们要证明可行性、实现一些小工具时，不论老手新手，用这样的方法准没错。但很可惜，这种方式并不适用于团队开发。

例如一个页面有三个页面模块，那么就至少会有3（个模块）×2（个类型：CSS/JS）=6个请求的产生，加上一些函数库，超过10个是轻而易举的：

```
<link href="bootstrap.min.css" rel="stylesheet">
<link href="base.css" rel="stylesheet">
<link href="_filter.css" rel="stylesheet">
<link href="_list.css" rel="stylesheet">
<link href="_info.css" rel="stylesheet">
<script src="jquery.js"></script>
<script src="_filter.js"></script>
<script src="_list.js"></script>
<script src="_info.js"></script>
<script src="base.js"></script>
```

传统加载模式问题为以下几点。

- 请求数量过多：适当的数量有助于并行下载，但过多则会造成性能问题。通常我们会鼓励一个网页的JavaScript与CSS的请求总数量维持在5个左右。
- 阻塞页面显示：若JavaScript或CSS文件所在服务器回应较慢，将会使得后面的HTML被延迟显示，对使用者浏览体验来说是很不好的。
- 内容未压缩：较专业的程序源代码在上线后需要压缩以加快下载速度。
- 在不同页面需顾虑前后顺序等问题：当页面较多时，我们可能得用后端的switch语句来决定载入哪些模块，这时常会有依赖关系造成前后顺序很难处理。

任何稍具规模网站采用这样的开发模式都不能视为专业。

页层级设定加载模式

页层级设定加载模式是指：“依不同的页面设定所需的JavaScript与CSS模块”。例如上面页面（称为Page A）设定要载入_filter.js、_list.js、_info.js、_filter.css、_list.css和_info.css等6个模块，而B页需载入_photo.css和_photo.js两个模块。所有的依赖模块都需要开发者在页层级的设定文件指定载入。

透过工具合.并及最小化档案

模块开发的主要问题在于它产生许多小的模块文件，得有一个工具或机制将这些零散且小的文件合并及最小化。本文以Mini（http://github.com/josephj/mini）工具为例，它有一个中央设定档：

```
<!-- A 页面 -->
<module id="page_a">
    <!-- 依赖模块档案 -->
    <file type="css" src="bootstrap.css">
    <file type="css" src="base.css">
    <file type="js" src="jquery.js">
    <file type="js" src="module.js">
    <file type="js" src="base.js">
    <!-- 页面模块档案 -->
    <file type="css" src="_filter.css">
    <file type="css" src="_list.css">
    <file type="css" src="_info.css">
    <file type="js" src="_filter.js">
    <file type="js" src="_list.js">
    <file type="js" src="_info.js">
</module>
<!-- B 页面 -->
<module id="page_b">
    <!-- 依赖模块档案 -->
    <file type="css" src="bootstrap.css">
    <file type="js" src="jquery.js">
    <file type="js" src="module.js">
    <!-- 页面模块档案 -->
    <file type="css" src="_photo.css">
    <file type="js" src="_photo.js">
</module>
```

你只要在浏览器输入以下网址http://localhost/mini?module=page_a&type=js，就可以得到page_a合并与最小化混淆过后的JavaScript文件。这样一下子就解决了传统加载模式中请求数量与内容未压缩的问题。

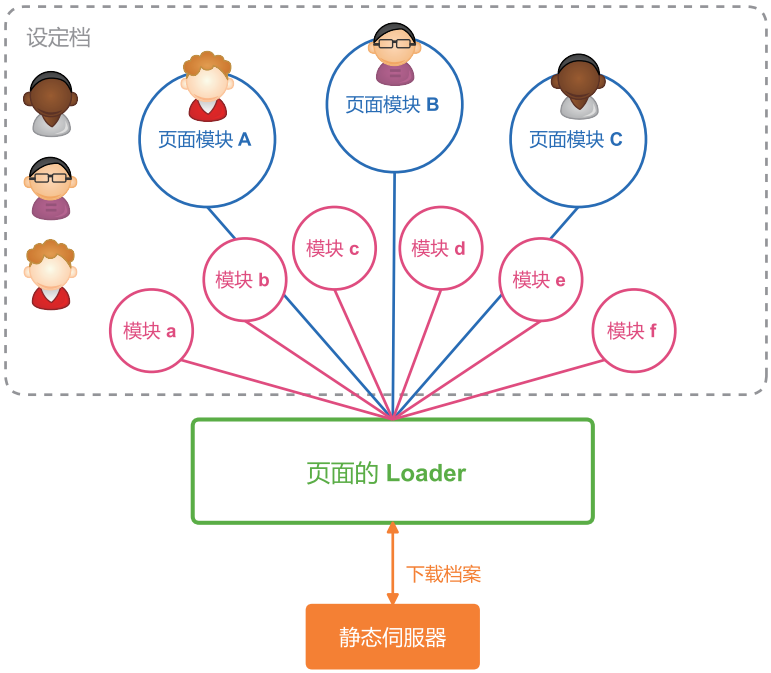


图3　页层级设定的架构示意图

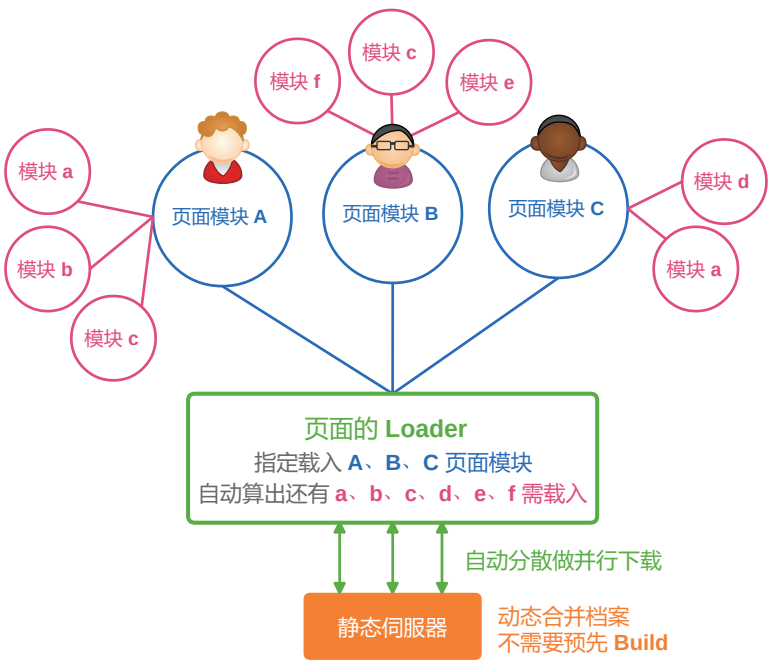


图4　模块层级设定的架构示意图

页层级设定的整体架构

图3中，虽然每位程序员专注于开发自己的模块，但都得共同修改同一个设定档，此设定档记载了每页要载入的模块有哪些。这是一个很凭直觉的作法。

页层级设定的缺点

- 维护不易：要移除一个页面模块时，当然要把依赖的模块也一并移除。但很不幸，实际的页面可能有10个以上的依赖模块，作为一个页面模块程序员，你并不能百分之百确认别的页面模块开发者会不会用到你所移除的依赖模块，因此常常会让模块越积越多。
- 发布耗时过久：这种页层级通常得在发布前预先做好合并及压缩，而当页面一多，整体的时间就会很久。根据持续集成的概念，网站Build的时间不应该超过10分钟，因此这一步通常会影响该目标。
- 线上调试困难：因为发布出去的都是混淆过后的代码，几乎无法做调试。

页层级设定最大的问题就是维护不易，主要是没有一个地方定义模块的依赖关系。此外我们既然已用页面模块作为开发单位，理应专注于模块的开发，不用管整个页面的复杂性，那为何还有整个页面的设定档呢？就模块化开发来说，页层级设定的思路并不理想。

模块层级设定加载模式

利用该模式，可以让我们向着真正的模块化开发前进一大步。

模块层级设定的整体架构

在模块层级的架构中（如图4所示），大致需要有3个必须的模式：

1. 程序员在此模式开始处要定义自身页面模块的依赖关系，图4的三个程序员各自开发不同的页面模块，而依赖的模块重复或不重复的都有。

2. 页面的Controller会指定要载入的页面模块有

```
<link rel="stylesheet" href="/combo/?g=css&f=stylesheets/_filter.css,stylesheets/_list.css,stylesheets/_info.css">
<script type="text/javascript" src="/combo/?g=js"></script>
<script type="text/javascript">YUI({"filter":"raw","combine":true,"comboBase":"\/combo\/?
f=","comboSep":",","logExclude":
{"Y.Module":true,"Y.ModuleManager":true},"root":"javascripts\/yui3\/build\/","langs":"en-US,zh-TW","groups":
{"demo":{"combine":true,"fetchCSS":false,"root":"","lang":null,"modules":{"_filter":
{"path":"javascripts\/_filter.js","requires":["module","node-base","cookie"],"type":"js"},"_list":
{"path":"javascripts\/_list.js","requires":["node-event-delegate","module","handlebars","cache","jsonp","node-
style","node-screen","event-resize","module","node-base","cookie"],"type":"js"},"_info":
{"path":"javascripts\/_info.js","requires":["module","node-base","handlebars"],"type":"js"}}},"module":
{"combine":true,"fetchCSS":true,"root":"javascripts\/module\/","lang":null,"modules":{"module":
{"path":"module.js","requires":["base","node-base","event-base","module-manager"],"type":"js"},"module-manager":
{"path":"module-manager.js","requires":["base"],"type":"js"}}}}}).use("_filter","_list","_info", function (Y)
{(new Y.ModuleManager()).startAll();});</script></head>
```

图5　使用StaticLoader Class输出复杂的<link/>、<script/>标签与YUI模块的设定

哪些，但不需提供依赖的那些模块，会由加载的Loader自动计算，得到所有应该要载入的模块。

3. Loader会与服务器端交互，透过一个聪明的机制（后文会详细解释）将模块分组并做合并与压缩，再用并行下载的方式将分组请求下载到浏览器上。

定义模块的依赖关系

前面开头我们提到AMD模块规范，其中很重要的设置便是定义每个模块的依赖关系，例如在下面这段代码中：

```
// AMD Module
define("editor", ["a","b","c"], function () {
    function Editor {
        /* Constructor */
    }
    return Editor;
});
require(["editor"], function (Editor) {
    new Editor();
});
```

- define定义了editor模块要载入a.js、b.js、c.js等三个模块。
- require在应用时只要指定editor，并不需要指定依赖的模块。

RequireJS是目前使用最为广泛的AMD架构实现。

让Loader载入相依的模块文件

以RequireJS来说，只要有定义好依赖关系，它就会依次将依赖的模块自动载入，并不需要任何配置。而当想要发布到线上时，你可以用Node.js所提供的r.js进行合并及最小化。

我们在页面层级设定时曾提到预先Build与最小化将会造成“发布时间过久”与“线上调适困难”两个问题，甚至在开发阶段，如果请求过多也将是个困扰，这都是RequireJS没办法解决的。我们必须寻找另外的解决方案。

YUI3——更好的模块加载方式

YUI3模块依赖关系设定与AMD如出一辙：

```
// YUI Module
YUI.add("editor", function () {
    function Editor {
        /* Constructor */
    }
    Y.Editor = Editor;
}, "VERSION", {requires:["a","b","c"]});
YUI.use("editor", function (Y) {
    new Y.Editor();
});
```

它的Loader下载方式让人眼前一亮，它使用一种称为Combo Handler的机制，将线上文件直接以GET的方式指定路径，动态地合并并最小化：

```
http://yui.yahooapis.com/combo?
    <模块1的对应路径>&
    <模块2的对应路径>&
    <模块3的对应路径>&
    ...
    <模块n的对应路径>
```

你可能会想到GET的长度限制，但聪明的YUI Loader早就考虑好了，它检查了“模块载入的先受顺序”、“模块总数量”、“目前浏览器的GET长度限制”、“浏览器同时检查数量”等信息，自动将向Combo Handler的请求分散为数个，且并行下载。

动手解决问题

YUI非常贴近我们的需求，但使用它，还是存在一些问题：

- Yahoo!的Combo Handler是否有开源或替代方案呢?
- YUI的组态设定非常复杂、不易维护：它们记载了模块依赖关系，但密密麻麻的可读性很差。
- CSS不适合做动态加载：大多数情况下，我们将CSS以<link/>的形式放在网页最前端，HTML一载入，浏览者就可以看到正确的UI，当然其中还有一些性能考量。但不论YUI还是RequireJS，它们都是动态加载的作法。

第一个问题很容易解决，网络上类似Combo Handler的机制还真的不少，像PHP5的Minify、Node.js的combohandler，稍微设置一下，很容易应用起来。

后面两个问题则需要写程序解决，我的解法是使用PHP完成一个称为StaticLoader的Class（https://github.com/josephj/static-loader）。它会读取模块的设定档：一边产出YUI那个复杂的组态档，一边针对CSS做特别处理，最后用link生成在页面上。

模块的设定档

所有的模块都集中在一个设定档，以下面的方式指定：

```
"_filter" => array(
  "group" => "demo",
  "js"    => "javascripts/_filter.js",
  "css"   => "stylesheets/_filter.css",
  "requires" => array(
    "module", "node-base", "cookie",
  ),
),
```

在这段代码中，你看到了JavaScript与CSS的路径，也指定了其依赖关系。

指定所要用的页面模块并且输出

在Controller中，只要指定这个页面有哪些页面模块，即可输出对应的link、script标签与模块依赖关系的设定（如图5所示）。

```
require_once "StaticLoader.php";
$loader = new StaticLoader("config.php");
$loader->set("_filter", "_list", "_info");
echo $loader->load();
```

之后就如我们所预期的，它将会把所有与此页面有依赖模块的JavaScript与CSS文件，以Combo Handler的方式，从服务器下载回来。

模块层级载入有什么好处

- Build更有效率：发布上线这一步我们不再需要处理CSS与JavaScript文件的合并与压缩，就可以直接将文件复制一份上传到服务器上，利用Combo Handler本身就具有的机制来做合并、压缩与缓存。
- 可以直接在线上做调试：原始文件皆存在，可以直接修正，只要把Minify对应的缓存清除即可。
- 维护性提升：开发者只定义自身模块的相依性，整页所需模块由YUI Loader计算后自动载入，管理容易许多，这才是真正的“模块化开发”。

结语

页层级的设置可想成“中央集权”——设置容易，初期一目了然，但规模变大后就不易管理。模块层级设定则是“地方自治”——设置较复杂，长期维护较有效率。配合相依性计算、自动合并、非同步下载等机制，可以让页面模块真正做到即插即用。

BigPipe是很多网站想实现的目标，但首先你得“建立模块相依性”、“采用页面模块开发模式”，建立模块导向开发模式才可能有实现的机会，我建议具有较大规模的网站不妨开始尝试朝着这个方向发展。

本文所涉及的完整案例，请参考http://josephj.com/lab/2012/github-chinese-developer/，完整的源代码可以从https://github.com/josephj/github-chinese-developer获得。



蒋定宇

首批加入台湾雅虎奇摩的Web Developer，主要负责项目的开发与前端架构。2009年加入创业团队miiCasa，目前担任前端技术团队主管，致力于持续地改善开发环境、工具、架构。

责任编辑：卢鹄翔（ludx@csdn.net）

SMSS离奇被杀之谜（上）

文 / 张银奎

多年来，我的办公桌前一直挂着一张半米高的钟馗画像。画中的钟馗手持利剑，二目圆睁，一派凛然正气。为什么挂这幅画呢？因为钟馗是驱邪捉鬼能手。办公室也有鬼吗？对于这个基本问题的任何简单回答都显得单纯无知。软件世界的鬼故事这就开讲。

F4崩溃

故事发生在某互联网服务商的机房中。机房建在偏远的郊区，远离都市的喧嚣。机房内层层叠叠摆放着各式各样的服务器，面板上的指示灯忽明忽暗。机房内很少有人出现，但并不安静，千奇百怪的风扇声音混杂在一处，噪声一片。朋友公司的服务器托管在这里很多年了。这些服务器曾经工作得非常稳定。只是最近一段时间才出了问题，隔三差五出现系统崩溃，以蓝屏形式宕机罢工。可能发生在白天，也可能发生在黑夜。

每次崩溃并重启后，通过查询系统日志（Event Viewer→Windows Logs→System）可以得到下面这样一条信息：

The computer has rebooted from a bugcheck. The bugcheck was: 0x000000f4 (0x0000000000000003, 0xfffffa8006a7db30, 0xfffffa8006a7de10, 0xfffff800037d3510). A dump was saved in: C:\Windows\MEMORY.DMP. Report Id: 091112-16723-01.

中间的一串数字便是所谓的蓝屏停止码和参数，开头的0x000000f4是停止码，括号里是4个参数。通常以停止码来对蓝屏崩溃进行简单分类，因此可以把这个蓝屏崩溃简称为F4崩溃。迄今为止，Windows系统共定义了将近300个停止码，分别代表不同情况的蓝屏崩溃。大多数停止码介于1和0x12C之间。只有少数几个停止码是超过3个16进制位的，例如0xDEADDEAD，它代表通过调试器故意触发的蓝屏。在Windows系统上工作这么多年后，我已对几十个停止码耳熟能详，其中就包括F4。熟悉这个停止码的一个原因是这个数字很响亮，让人想起著名的F4战斗机，还有我读书时曾经流行的一个歌手组合。

搜索或者查阅WinDBG的帮助手册，都可以找到关于F4停止码的详细描述。长话短说，F4代表关键对象终止（CRITICAL_OBJECT_TERMINATION），也就是系统的关键进程或者线程意外结束。第一个参数代表对象的类型，3代表进程，6代表线程；第二个参数是被终止对象的地址；第三个参数是进程的映像文件名称地址；第4个参数是包含解释性信息的ASCII字符串地址。

至此，根据日志信息中的停止码和第一个参数，可以知道导致蓝屏的原因是某个进程意外终止了。因为后面几个参数都是地址值，需要读取其中的内容才能知道进一步的信息，系统已经重启了，如何读取这些地址的内容呢？日

志的倒数第二句话提示我们，系统保存了一个转储文件（C:\Windows\MEMORY.DMP）供我们进一步分析。

何人被杀

通过网络将服务器上的转储文件复制到本地，然后唤出WinDBG，打开转储文件。WinDBG做了一番自动分析之后，显示出命令提示符等待更多命令。首先我们迫切想知道的当然是被杀进程的名称，因为停止码的第三个参数指向的便是被杀进程的映像文件名称（字符串），因此执行一下da命令便可以了：

```
0: kd> da 0xfffffa8006a7de10

0xfffffa8006a7de10 "smss.exe"
```

哦，是smss！虽然整个名字都是小写，好像不起眼，但我每次遇到它，都是“刮目相看”。而我每次写这个名字，都用大写字母SMSS以示重视。打个比方，对熟悉Windows操作系统的人来说，SMSS就像追星的少男少女眼里的大S和小S那样有名。

这么说一点不夸张。不妨略述两点理由：首先，从Windows系统启动过程来看，SMSS是第一个完完全全的进程，此前的IDLE进程和系统进程都只有内核态空间，没有用户态空间，是杜撰出来的。而SMSS是第一个通过执行EXE方式创建的完整进程。系统内核初始化好内核空间后，创建这个进程来开创用户态世界。SMSS启动后，创建Windows子系统服务器进程CSRSS和负责登录的WinLogon进程，后者再创建负责安全认证的LSASS进程、管理后台服务的services.exe，以及登录桌面。从进程的父子关系来看，SMSS可谓是所有其他用户态进程的鼻祖。另外，SMSS的全名叫会话管理器子系统（Session Manager Subsystem），它掌管着系统“人机会话”大权。

简单说，一个会话就是具有输入输出的交互系统。今天的Windows系统中一般至少有两个会话，一个是运行后台服务的会话，另一个是运行普通应用程序的普通会话。如果远程登录或者多用户登录，那么需要向SMSS申请创建更多个会话。

如此说来，SMSS在Windows系统中真可谓是位高权重，具有特殊的尊崇地位，在很多方面都享有特殊待遇。就拿眼下的这个崩溃为例，也是因为内核发现被杀的是SMSS进程而将整个系统叫停。如果被杀的是普通进程，那么内核才不会这么做，杀就杀了。

为了保护SMSS，系统做了很多工作，例如在Windows Vista之前，使用任务管理器是不能杀掉SMSS进程的，如果这么做，会给出一个图1所示的对话框。

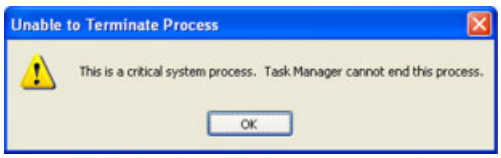


图1 任务管理器不杀SMSS

从Windows Vista开始，如果没有管理员权限，那么在任务管理器中根本就看不到SMSS的尊荣。如果以管理员身份运行任务管理器，并试图终止SMSS，会得到一个看起来很离奇的对话框（如图2所示），直接问你是不是要关机。在认识到后果的严重性，并承诺放弃尚未保存的数据后，关机按钮（Shut down）加亮。点击之后，便立即蓝屏了，绝对比普通的关机要快很多倍。

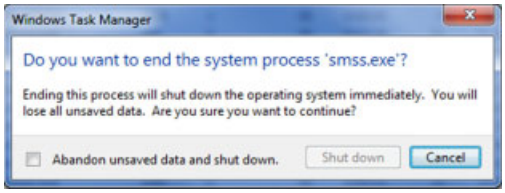


图2 别样的关机方式

杀手是谁

既然SMSS地位不凡，被层层保护，那么现在是如何被杀的呢？杀手又是谁呢？不妨先执行k命令看一下栈回溯，结果如图3所示。

好清晰的函数调用过程，最上面的#00栈帧是著名的蓝屏函数KeBugCheckEx。它是蓝屏崩溃过程的入口，一旦进入这个函数，系统便走上了不归之路，有去无回。#01栈帧的PspCatchCriticalBreak是进程管理器的一个内部函数，专门用来嗅探关键“人物”被杀。发现这样

的行为后便调用蓝屏函数，拉响警报。#02栈帧的函数名应该是PspTerminateAllThreads，因为二进制优化的缘故，WinDBG没能找到确切的函数名称。#03栈帧是系统服务NtTerminateProcess，是用来结束进程的公开服务，也正是用户态的TerminateProcess API的底层实现。#04栈帧系统服务的分发函数。从地址和模块名都可以看到，#05栈帧属于用户态。

这样一路追溯下来，可以明显看出是用户态有人在调用终止进程的系统服务来杀SMSS。那么，是谁发起的这个调用呢？有必要看一下当前进程是谁。执行!process命令可以给出当前进程的基本信息，如图4所示（为节约篇幅，这里只显示前半部分）。

结果的第1行是进程的EPROCESS结构地址，从这个地址可以看到系统是64位的。第2行信息的含义依次是：进程属于会话1，进程的ID是0xC2C，进程的用户态环境块地址，以及进程的父进程ID。跳过第3行，直接到第4行观察映像名称（Image），赫然写着taskmgr.exe，也就是任务管理器。这个结果很让人吃惊。

怎么会是任务管理器呢？如我们刚才所讲，任务管理器非常清楚SMSS的身份，是不会轻举妄动来杀SMSS的，除非有管理员让它这么做，这是个服务器系统啊，我已经与朋友反复确认没有用户在本地操作。

但调试器不会说谎，当前进程确实是在执行杀戮SMSS的系统调用，而且当前进程也确实就是任务管理器。任务管理器为什么会执行这个大逆不道的操作呢？它想要起义造反，还是被鬼使神差了呢？

作案过程

前面分析使用的都是内核转储文件，不包含用户空间的数据，因此在图3所示的栈回溯中，我们只能看到内核态的执行过程，没有看到用户态中是如何发起调用这个系统服务的。为了能观察到用户态的过程，特意让朋友将系统设置改为完全转储。这需要先将注册表的HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\

```
0: kd> kn
 # Child-SP          RetAddr           Call Site
00 fffff880`05eb7ac8 fffff800`01a57892 nt!KeBugCheckEx
01 fffff880`05eb7ad0 fffff800`01a03e8b nt!PspCatchCriticalBreak+0x92
02 fffff880`05eb7b10 fffff800`01982f74 nt! ?? ::NNGAKEGL::`string'+0x176d6
03 fffff880`05eb7b60 fffff800`016cf453 nt!NtTerminateProcess+0xf4
04 fffff880`05eb7be0 00000000`774915da nt!KiSystemServiceCopyEnd+0x13
05 00000000`0025ebe8 00000000`00000000 ntdll!NtTerminateProcess+0xa
```

图3 杀戮现场

```
0: kd> !process
PROCESS fffffa80057b4060
    SessionId: 1  Cid: 0c2c    Peb: 7ffffdf000  ParentCid: 0840
    DirBase: 89e13000  ObjectTable: fffff8a008cc5d70  HandleCount: 127.
    Image: taskmgr.exe
    VadRoot fffffa8005a0ec90 Vads 99 Clone 0 Private 769. Modified 276968. Locked 0.
    DeviceMap fffff8a0015f35f0
    Token                             fffff8a008869a90
    ElapsedTime                       01:26:27.627
    UserTime                          00:00:00.031
    KernelTime                        00:00:01.950
    QuotaPoolUsage[PagedPool]         0
    QuotaPoolUsage[NonPagedPool]      0
    Working Set Sizes (now,min,max)  (2828, 50, 345) (11312KB, 200KB, 1380KB)
    PeakWorkingSetSize                2828
    VirtualSize                       85 Mb
    PeakVirtualSize                   144 Mb
    PageFaultCount                    5320
    MemoryPriority                    BACKGROUND
    BasePriority                      13
    CommitCharge                      926
```

图4 杀手档案

CrashControl表键下的CrashDumpEnabled键值设置为1，然后重启系统。

改变设置并且过了一段时间后，崩溃又发生了。完整转储文件也抓到了，大小超过4GB，压缩后大约500MB。感谢现今的网络速度，文件没费什么力气就从机房经朋友的公司“转机”到了上海。

将压缩文件释放，在WinDBG中打开。仍然是F4停止码，仍然是SMSS被杀，仍然发生在taskmgr进程之中。

因为系统中有很多进程，每个进程都有自己的用户态空间，所以在分析系统转储时观察用户态栈回溯时，一定要注意切换进程，要确保将要观察的进程切换当前进程。但对于我们眼下的问题，当前进程就是taskmgr进程，因此不再需要切换，但仍然要让WinDBG来刷新用户态模块列表。这可以通过执行.reload /user命令来完成。

```
1: kd> .reload /user

Loading User Symbols
```

刷新完毕后，执行kn命令观察栈回溯，结果如图5所示。

感谢WinDBG给出了一个完美的栈回溯结果，把当前线程从用户态到内核态的整个执行过程都呈现在我们面前。根据这个栈回溯，我们可以很轻松地观察到SMSS被杀戮的全过程。#12栈帧是任务管理器的主函数wWinMain，#11和#10栈帧是典型的消息循环函数（使用对话框作为主窗口）。#0f-#0b栈帧是将消息逐级分发给窗口函数，从

```
1: kd> kn
 # Child-SP          RetAddr           Call Site
00 fffff880`06034ac8 fffff800`01a24892 nt!KeBugCheckEx
01 fffff880`06034ad0 fffff800`019d0e8b nt!PspCatchCriticalBreak+0x92
02 fffff880`06034b10 fffff800`0194ff74 nt! ?? ::NNGAKEGL::`string'+0x176d6
03 fffff880`06034b60 fffff800`0169c453 nt!NtTerminateProcess+0xf4
04 fffff880`06034be0 00000000`777515da nt!KiSystemServiceCopyEnd+0x13
05 00000000`0021ebd8 000007fe`fd9e34ff ntdll!NtTerminateProcess+0xa
06 00000000`0021ebe0 00000000`ff5b6904 KERNELBASE!TerminateProcess+0x2f
07 00000000`0021ec10 00000000`ff5b8d08 taskmgr!CProcPage::KillProcess+0x12c
08 00000000`0021ec70 00000000`ff5b8efc taskmgr!CProcPage::KillAllChildren+0x98
09 00000000`0021ecf0 00000000`ff5b7c58 taskmgr!CProcPage::RecursiveKill+0x170
0a 00000000`0021f1a0 00000000`ff5b81d8 taskmgr!CProcPage::HandleWMCOMMAND+0x318
0b 00000000`0021f220 00000000`776279b7 taskmgr!ProcPageProc+0x47c
0c 00000000`0021f2f0 00000000`77627792 USER32!UserCallDlgProcCheckWow+0x1b6
0d 00000000`0021f3b0 00000000`776276c2 USER32!DefDlgProcWorker+0xf1
0e 00000000`0021f430 00000000`77619bd1 USER32!DefDlgProcW+0x36
0f 00000000`0021f470 00000000`776198da USER32!UserCallWinProcCheckWow+0x1ad
10 00000000`0021f530 00000000`776167c2 USER32!DispatchMessageWorker+0x3b5
11 00000000`0021f5b0 00000000`ff5aab0d USER32!IsDialogMessageW+0x153
12 00000000`0021f640 00000000`ff5c2a2a taskmgr!wWinMain+0x921
13 00000000`0021fc90 00000000`774f652d taskmgr!DelayLoadFailureHook+0x206
```

图5 完整的杀戮过程

```
mov     rcx,qword ptr [taskmgr!g_hMainWnd (00000000`ff5ca358)]
lea     rdx,[rsp+58h]
call    qword ptr [taskmgr!_imp_IsDialogMessageW (00000000`ff5a1598)]
cmp     eax,r13d
jne     taskmgr!wWinMain+0x93c (00000000`ff5aab28)
```

图6 宝贵的线索

#0a栈帧的方法名HandleWMCOMMAND可以推测分发的是一个WM_COMMAND消息。

接下来的三个栈帧都杀气腾腾：RecursiveKill（递归方式杀）、KillAllChildren（杀所有子进程）和KillProcess（杀某一个）。

而后便是调用TerminateProcess API，后者再调用系统服务而进入到内核去执行了。

这个栈回溯结果包含了丰富的信息，给了我们很多新证据，同时也带给我们很多疑问。宏观上看，整个执行过程是的典型窗口消息处理过程，层层相扣，严丝合缝。参与执行过程中的模块也很少，除了taskmgr外，都是著名的系统模块，Windows子系统的接口模块user32、内核API的接口模块kernelbase、具有“内核血统”的ntdll和内核本身。taskmgr也是Windows自家的，因此都是“自己人”，没有第三方模块介入和干涉。

神秘消息

从图5所示的执行过程看不出什么破绽，“顺理成章”，每一环都很严谨。过程没有问题，那么问题可能出在“外部输入”上。到底是收到了一个什么样的消息才使任务管理器开始执行这个杀戮过程的呢？

想找到这个消息需要费些功夫。如果是32位系统，那么可以比较容易地通过摆在栈上的参数寻找到。但现在分析的是64位系统，参数主要是通过寄存器来传递的，前4个参数通常使用RCX、RDX、R8和R9来传递，更多的参数才放在栈上。寄存器值是很容易变化的，因此调试64位系统时，追溯参数值变得比以前要困难得多。如何才能解决这个问题呢？第一种方法是修改被调试模块的编译选项，让编译器产生代码将用寄存器传递的参数也复制到栈上一份。第二种方法是分析函数的代码，看函数内部是否曾将参数赋值给局部变量或其他变量，然后再观察这些变量的值，至于函数会不会做这个复制操作，那就要看具体情况了。

对我们来说，第一种方法根本不适用，因此只好尝试第二种方法了。功夫不负有心人，经过一番分析，还真的找到了突破口。

#11栈帧是公开的IsDialogMessage函数，SDK中有它的函数原型：

```
BOOL IsDialogMessage(
                    HWND hDlg,
                    LPMSG lpMsg
                    );
```

它的第二个参数是消息结构，我们无论是在IsDialogMessage函数内部还是在它的父函数wWinMain中找到这个参数的取值都可以。观察wWinMain中调用IsDialogMessage函数时的代码（如图6所示），让我们欣喜若狂。

图6中的几句汇编很容易理解，先将全局变量g_hMainWnd的值赋给RCX寄存器，也就是在准备调用IsDialogMessage的第一个参数。接下来将RSP+58作为有效地址赋值给RDX，作为第二个参数，第二个参数就是消息结构的地址啊，这样看来，RSP+58处放的就是消息结构。RSP是栈指针，在64位Windows使用的调用协议中，它同时又是被用做索引局部变量栈指针。栈回溯时可以回溯出每个函数的RSP值，上面kn结果中的第1列便是这个值。

如此看来，从kn结果中找到wWinMain这一帧的RSP值，然后加上58便是我们梦寐以求的消息结

```
1: kd> dd 0021f640+58
00000000`0021f698  0004066c 00000000 00000111 00000000
00000000`0021f6a8  00009ca6 00000000 00000000 00000000
00000000`0021f6b8  058e614e 00000049 0000020b 00000000
00000000`0021f6c8  0021f990 00000000 0000408b 00000000
00000000`0021f6d8  ff5a7058 00000000 00000000 00000008
00000000`0021f6e8  ff5a0000 00000000 00000000 00000000
00000000`0021f6f8  00010003 00000000 00000000 00000000
00000000`0021f708  00000000 00000000 ff5a69d8 00000000
```

图7 触发杀戮的窗口消息

表1 WM_COMMAND消息

Message Source	wParam (high word)	wParam (low word)	lParam
Menu	0	Menu identifier (IDM_*)	0
Accelerator	1	Accelerator identifier (IDM_*)	0
Control	Control-defined notification code	Control identifier	Handle to the control window

构地址了。找到前者的值是0021f640，赶紧使用dd命令观察（如图7所示）。

因为没有使用的公开调试符号中缺少WND结构的类型信息，所以我们只能直接观察内存数据，并根据SDK中的结构定义来核对每个字段的值。

```
typedef struct {
  HWND hwnd;
  UINT message;
  WPARAM wParam;
  LPARAM lParam;
  DWORD time;
  POINT pt;
} MSG, *PMSG;
```

根据上面的结构定义，第一个字段是窗口句柄，64位程序是64位宽，结合图7，它的值是00000000`0004066c，根据经验，这个值像是窗口句柄，但还需要其他证据来帮助确认。第二个字段是消息ID，值为111，根据记忆，这正是WM_COMMAND常量的值，保险起见，找到头文件中的定义，果然如此。

```
#define WM_COMMAND      0x0111
```

这证明我们找到的消息结构是可靠的。按照这样的方式进行考证，我们可以复原出消息结构的完整内容：

```
HWND hwnd = 0004066c;
UINT message = 0x0111; // WM_COMMAND
WPARAM wParam = 00009ca6;
LPARAM lParam = 0;
DWORD time = 058e614e;
POINT pt = { 00000049, 0000020b };
```

有了数据，我们再看SDK中关于WM_COMMAND消息的描述来理解这个消息的深层信息。WM_COMMAND是个比较复杂的消息，为了将其描述清楚，SDK特别画了一个表格（如表1所示）。

将我们的数据与表格描述一一对应，可得到如下两条推论。

- 这个消息来自菜单，因为lParam=0，而且wParam的高地址字段为0。
- 菜单消息的ID是00009ca6。

根据这个推论，任务管理器是因为收到了命令号为00009ca6的菜单命令而执行杀戮动作的。

模拟作案过程

电视中警察破案时，经常实地模拟可能的作案过程，用来核实推论和启发思路。我们不妨也用用这招。以管理员身份调出任务管理器，切换到进程表单，将SMSS进程选中，然后点击鼠标右键，调出的快捷菜单中有两项用来终止进程的，End Process和End Process Tree，根据我们前面的分析，显然后者更接近我们分析的情况。点击这一项，得到确认对话框。

对话框中的文字反复强调了继续操作可能导致的严重后果，可能丢失数据，可能导致整个系统不可用。确认系统中没有什么重要的程序还在运行后，点击End process tree，屏幕立刻一闪，然后一片蔚蓝，之后自动重启了。重启后分析转储文件，与我们上面分析的如出一辙。

这样分析过后，明显就是有人在任务管理器里点击右键，选择End Process Tree，再沉着冷静地在确认对话框中再次选择End process tree按钮而把SMSS杀掉的。于是打电话给朋友，反复询问他们是不是在开玩笑，是不是有哪个操作员在搞恶作剧或者误操作了。朋友的回答很肯定："这个问题折磨我们几个月了，反复出现了无数次，能登录服务器的几个人都知道这个问题，怎么可能是我们自己人点的呢？"没人点，任务管理器从哪里收到一条神秘的杀进程消息呢？机房里真有鬼不成？欲知黑手来自何方，下期将继续探索。



张银奎

《软件调试》作者，从事软件开发和研究十余年，对IA-32架构、操作系统内核、虚拟技术，尤其对软件调试有较深入的研究。

责任编辑：卢鹄翔（ludx@csdn.net）

有奖问答：

上期问题：关键区结构中的SpinCount字段的含义。当这个字段不为0时，如果某个线程想进入关键区，而且里面已经有"人"，那么这个线程会循环等待里面的"人"出来，每循环一次，查询一下，直到循环SpinCount值指定的那么多次。这样做的好处是如果循环一定次数后就能进入关键区，那么就免去了调用等待函数而进入内核态被挂起，可以提高程序执行速度和关键区的效率。

本期问题：完整转储文件（Complete memory dump）与内核转储文件（Kernel memory dump）的主要差异是什么？

编者说明：

- 编辑信箱：ludx@csdn.net
- 联系方式写在一个单独的TXT文件里，包括以下几项：
 1）姓名
 2）工作单位或学校
 3）电话联系方式
 4）邮寄地址
 5）E-mail地址
- 解答提交时间，最好早于当月15日。

名人堂

小鳄鱼之父

Bart Decrem



文 / 亦然

迪士尼独立开发的游戏《Where is my water》从2011年9月推出后在两个月内就赢得了79个国家App Store付费游戏排行榜冠军。小鳄鱼Swampy一举成为迪士尼在移动世界的新代言人，除了“小鳄鱼”外，此后与Imangi合作的《Temple Run Brave》也取得了不错的成绩，一度有三款来自迪士尼的游戏占据了App Store付费榜前十的席位。它们也帮助迪士尼的移动游戏部门从被人遗忘的角落走到了舞台的中心地带，而这背后的推手是一位颇具传奇色彩的人物——Bart Decrem。

Bart有着极为丰富的创业经历，他出生并成长于比利时，在斯坦福大学获得法学博士学位。自古英雄出少年，13岁时，Bart就开始在一家地下电台成为了一名DJ，并创办了该电台的周刊杂志。1999年，他和朋友合伙创建了Eazel公司，主营开源的桌面Linux系统。这家公司开发了大名鼎鼎的Nautilus——一款在Linux系统下Gnome桌面环境中重要的文件管理器。但由于诸多原因，这家公司于2010年倒闭。

2003起，Bart开始在Mozilla主导市场营销及商务工作，负责Firefox 1.0的发行推广及商务事宜。2004年，Bart离开Mozilla后创建Flock公司并出任公司CEO，开发了同名产品Flock，这是一款基于Chromium的社交化浏览器，2011年该公司被Zynga收购，Flock浏览器停止服务。而在这期间，Bart其实已经离开Flock，于2007年就职于Doll资本管理公司 。2008年与Andrew Lacy创建了Tapulous，iOS畅销音乐游戏《Tap Tap Revenge》就是Tapulous的产品。2009年5月，他被美国知名商业媒体Fast Company誉为商界100位最具创新人物。

2010年7月，迪士尼以5.6亿美元收购Tapulous后，在其基础上设立了移动部门，Bart开始担任迪士尼移动部门高级副总裁兼总经理，负责迪士尼全线移动游戏业务，他的首要目标就是帮助迪士尼在移动互联网创造一个新的动画明星。随后，小鳄鱼Swampy肩负了这一重任。但Swampy的形象本身实际上并不像米老鼠那样简单且具有传播力，好在Bart他们想出了一个不错的解决方案：为游戏添加丰富的故事场景，帮助Swampy更久地活跃于玩家的脑海之中。

“其实角色并不是最重要的”，Bart曾在一次采访中谈道他对游戏设计的看法：“我们首先确定了游戏的玩法机制。优先考虑的是能否制作出让人们玩上很久的游戏，能否创建一个有生命力可扩展成迪士尼代表的游戏。”

如今，如Bart所期望的那样，小鳄鱼系列已经成长为一个家族：一部以Swampy为主角的12集动画短片将在迪士尼官方网站和YouTube上播出，游戏中的其他角色也会有相应的产品推出……

对于个人开发者和小的开发团队如何做出畅销游戏，Bart给出的建议是：“App Store是个和传统行业完全不同的地方，这里没有竞价排名，苹果也会去寻找真正优秀的产品进行推荐。《Tiny Wing》出自一位德国开发者之手，他基本没有做过推广，却取得了骄人的成绩。还有一些离开了Tapulous自己开发游戏的开发者，同样取得了非常好的成绩。他们根本没有所谓的长期战略或者独到的眼光，只是‘在做让自己觉得满意和喜爱的东西’”。

推荐图书

云计算时代，我与Go语言难解的缘分

文 / 许式伟



Go语言编程

作者：许式伟，吕桂华等

我在新浪微博、《Go语言编程》一书中下了一个论断：Go语言将超过C、Java，成为未来十年最流行的语言。

我曾在2007年第2届ECUG大会上提到了我对未来软件产业趋势的判断：存储与计算将向服务端转移；PC单机转换为强悍的服务器＋多元化的终端。这个趋势判断对我的职业生涯其实影响非常重大。

从业务角度讲，正因为这个判断，促使我选择成立金山实验室研究云存储技术；也促使我关注Erlang这门面向分布式编程的领域性语言；促使我举办了一届届的ECUG大会；促使我加入盛大创新院，发起祥云计划（盛大云前身），做盛大网盘；促使我们成立了七牛，有了七牛云存储。

从技术角度来讲，它促使我放弃了近10年的桌面开发经验，转向服务端开发。正如我在《我为什么选择了Erlang》中建议的那样：

- 要么不写程序，要么就写服务器端程序；
- 当然，你也可以去撰写移动终端设备上的代码，在PC平台上做开发的空间很小。

在金山实验室时，我选择了Java语言，并积极关注Erlang语言，对其进行小范围试用，以及大规模使用的可行性评估。我的结论是：

- Erlang风格的并行思想真的很棒；
- Erlang语言优势与劣势同样明显，这门语言不适合规模化推广使用。

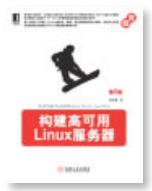
Erlang的困难之处在于它是FP语言。我们缺乏深入人心的FP编程理论。我们并不了解FP“数据结构”学。这是Erlang语言无法逾越的门槛，决定了它只能是小众化语言。

进入盛大创新院后，我选择了C++，原因是我个人更喜欢C++，而对Java的设计哲学非常不感兴趣。出于对Erlang风格的并行思想的喜爱，我创建了一个名为CERL的项目，取义Erlang Model for C++，表明我希望能够把Erlang那一套并行编程模型搬到C++中来，降低分布式编程的负担。

接触Go语言后，我发现CERL 2.0的并发编程模型，其实就是一个雏形版的Go语言并发模型，两者从理念上来说完全一致。

七牛公司成立于2011年5月底，我们选择了Go语言作为七牛云存储服务端的主体语言。当时Go语言甚至语法都还没完全稳定下来。为什么我可以如此坚定地相信，选择Go语言不会有错，并且相信Go语言会成为未来十年最流行的语言？除了Go语言的并发编程模型深得我心外，Go语言的各种语法特性显得那么深思熟虑、卓绝不凡，其对软件系统架构的领悟，处处带给我惊喜。

本月新书



构建高可用Linux服务器（第2版）

作者：余洪春

第2版在第1版的基础上进行了修订和补充，补充了大量关于高可用的内容，提高了本书的含金量。对不同层次的运维人员有不同层次的帮助。对新手来说，这本书就像引路人，从系统安装一直到服务器集群高可用的构建，非常广泛地介绍了运维的相关工作。同时，作者结合自己的经历和经验给出了一些有指导意义的建议，不仅能帮新手提高技能，在职业规划方面给予了一定的启发。对有一定经验的运维人员来说，书中的运维经验及思想非常值得借鉴。



程序员的职业素养

作者：Robert C.Martin

译者：章显洲，余晟

软件从业人员可能经常会遇到项目进度拖延、修复各种Bug、做各种测试等情况。但能否‘静下心来想想，为何拖延项目进度？是因为产品需求的频繁改动，还是高估了自己的能力，或者受到了干扰？为何自己编写出来的代码总是Bug不断？阅读完Bob大师的这本书，所有问题都迎刃而解了。作者40多年的编程生涯心得体会，会让你自己的职业生涯有所顿悟。虽然本书讲的是程序员的职业素养，但也能帮助测试工程师、系统工程师等相关人员培养自己的职业素养。

Go语言给我的第一个意外惊喜，是接口。当然，我意外的不是Go的非侵入式接口。在接口（或契约）的表达上，我一直认为Java和C#这些主流的静态类型语言都走错了方向。C++的模板尽管机制复杂，但走在了正确的方向上。但Go语言的接口远不是非侵入式接口那么简单，它是Go语言类型系统的纲。除了支持反射等高级特性外，Go语言还支持接口查询。

Go语言给我的第二个意外惊喜，是极度简化但完备的“面向对象编程”方法。Go语言废弃大量的OOP特性，如继承、构造/析构函数、虚函数、函数重载、默认参数等；简化的符号访问权限控制、将隐藏的this指针改为显式定义的receiver对象。让我看到了OOP编程核心价值原来如此简单——只是多数人都无法看透。

Go语言带给我的第三个惊喜，是它的函数多反回值和错误处理规范。函数多返回值比较容易想到，只有这样函数的输入输出才能清晰呈现，语义表达上才会足够清晰。不过让我没想到的是，Go引入了内置的error类型以及defer关键字来编写异常安全代码，让人拍案叫绝。

Go语言带给我的第四个惊喜，是它功能的内聚。一个最典型的案例是Go语言的组合功能。对于多数语言来说，组合只是形成复合类型的基本手段。但Go语言引入了匿名组合的概念，让其他语言原本需要引入继承甚至虚拟继承这样晦涩概念来完成的事情，统一到了组合这样的一个基础上。

Go语言带给我的第五个惊喜，是消除了堆与栈的边界。在Go语言之前，程序员是清楚地知道哪些变量在栈上，哪些变量在堆上。堆与栈是基于现代计算机系统的基础工作模型上形成的概念，Go语言屏蔽了变量定义在堆还是栈上这样的物理结构，相当于封装了一个新的计算机工作模型。这一点看似与Go语言显式表达的设计哲学不太一致，但我认为这是一项了不起的工作，且与Go语言的显式表达并不矛盾。Go语言强调的是对开发者的程序逻辑（语义）的显式表达，而非对计算机硬件结构的显示表达。对计算机硬件结构的高度抽象，将更有助于Go语言适应计算机硬件发展的变化。

Go语言带给我的第六个惊喜，是Go语言对C语言的支持。可以这么说，Go语言是除了Objective-C、C++这两门以兼容C为基础目标的语言外的所有语言中，对C语言支持最友善的一个。什么语言可以直接嵌入C代码？没有，除了Go。什么语言可以无缝调用C函数？没有，除了Go。对C语言的完美支持，是Go快速崛起的关键支撑。还有比C语言更让人觊觎的社区财富吗？那是一个取之不尽的金矿。

在《Go语言编程》一书的前言中，我谈到Go语言的基础哲学来源于C语言，而不是像Java、C#那样，学的是C++。C语言的成功在于它足够简单，所以Go语言也要非常简单。Go语言的定位就是成为互联网时代的C语言。本质上来说，Go语言的特性都围绕着以下设计哲学：

- 大道至简，显式表达。任何封装都是有漏洞的。最佳的表达方式就是最直白的表达方式，不试图去做任何包装。
- 最少特性原则。如果一个功能不对解决任何问题有显著价值，那么就不提供。

尽管是40年来出现的语言非常之多，但我认为，谈得上突破了C语言思想，将编程理念提高到一个新高度的，仅有Go语言而已。

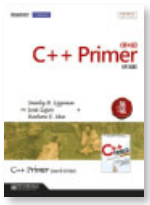
Go语言是一门变革性的语言，要革的是C的命（顺便革Java的命）。Go语言很简单，但具备极强的表现力。从目前的状态来说，Go语言主要关注服务器领域的开发，但这不是Go语言的完整使命。

我们说Go语言适合服务端开发，仅是因为它的标准库支持方面，目前是向服务端开发倾斜：

- 网络库（包括 Socket、HTTP、RPC 等）；
- 编码库（包括 JSON、XML、GOB等）；
- 加密库（各种加密算法、摘要算法）；
- Web（包括 Template、HTML支持）。

而作为桌面开发的常规组件GDI和UI系统与事件处理，基本没有涉及。

总之，我认为，Go语言将引领未来10年IT产业的发展。在最初5年内，Go语言会在服务器端编程上大放异彩，而桌面端的开发则仍然处于探索和完善期，预计在后5年才趋于成熟，成为各种手持设备上的主流开发语言之一。

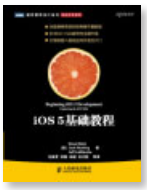


C++Primer（第4版）（评注版）

作者：Stanley B.Lippman　Josee Lajoie　Barbara E.Moo

译者：陈硕

一本久负盛名的C++经典教程，全面而深入地讲解了C++语言及其标准库。本书从为什么要学习C++开始讲起，通过举例子的方式，从语言特性、开发成本、系统性能等方面诠释C++语言的精髓。本评注版由国内资深专家陈硕执笔，在英文原著基础上增加中文点评与注释，既保留经典的原创文字与味道，又以先行者的学研心得与实践感悟，帮助读者阅读和理解。经过评注的版本，值得反复阅读与体会。同时，本书中的代码质量很高，也很有利于读者学习和使用。



iOS 5基础教程

作者：Mark,D.　Nutting,J.　LaMarche,J.

译者：毛姝雯，漆振，杨越，孙文磊

本书是一部关于iOS SDK开发的基础教程，内容翔实、语言生动。几位作者结合消费类设备上的常见实例，循序渐进地讲解了适用于iPhone 4S、iPad 2及iPod Touch开发的基本流程。书中介绍了强大的iOS 5操作系统，涵盖xcode 4以来的新功能，其中最为引人注目的便是以全新章节详细介绍Storyboard 和iCloud。全书所有项目均使用xcode 4.2 重新创建，让开发者全面感受xcode 4带来的振奋人心的新变化。同时，本书具有较强的通用性，各层次读者都能通过本书快速学习iOS 开发。

GEEK产品



Wheelman - Motorized Skateboard

这个很酷的电动滑轮车，轮胎防滑性极佳，拥有手持式油门和前脚控制的刹车系统，能够承载90公斤的车手以32/h的速度跨越各种地形。

StethoCloud

澳大利亚墨尔本大学的几位科学家开发出一套叫做“StethoCloud”的软件。基于云计算技术，这套软件可以在医生不在场的情况下通过智能手机对肺炎进行初步诊断。用户先要将“听诊麦”连接在手机的耳机插口上。随后，StethoCloud会指导用户操作“听诊麦”，同时听取并记录病人的呼吸音。当软件获取到足够的数据后，它会自动将数据上传至位于云端的服务器来消除噪音并进行分析。分析完成后，服务器会将诊断结果与治疗建议一起发回到用户的手机上。







Owl 360 - Rear View Bicycle Camera

这款自行车拥有独特的后视摄像头系统，只需将相机安装在自行车的座位后，并把彩色显示器安装在把手上，就可以帮助骑行的人不必回头也能看见后方的状况。

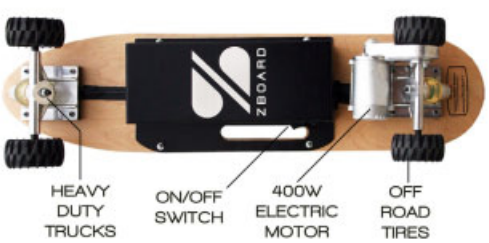
App-Controlled Wireless Spy Tank

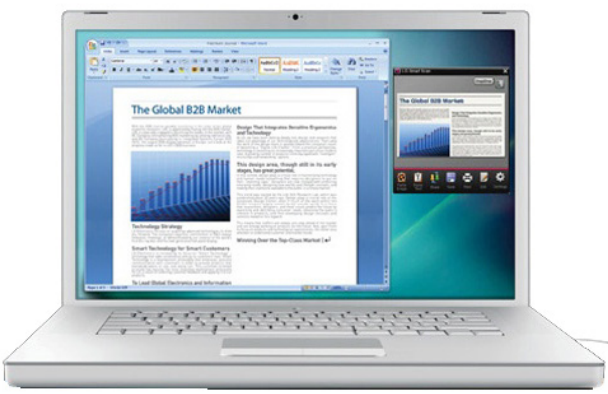
这个无线间谍坦克可以通过智能手机或平板电脑控制，它拥有加速度计、夜视摄像头、麦克风、扬声器和一个广角镜头。你可以控制它到60米远的地方，用它录音、拍摄照片或视频。

Zboard

这款Boosted Boards公司推出的电池自动推进式长板电动长板号称是史上最轻的电动交通工具，最高速度可达32.2千米/小时，电池续航可以让长板行驶约9.7千米。长板还配备了手持无线控制器来控制速度，使用标准的充电接口，两小时就能充满电。







扫描鼠标LSM-100

LG推出的智能扫描鼠标LSM-100与普通鼠标相比，在左侧的边缘多了两个按钮，轻按上方的Scan按钮可以扫描任何A3尺寸的文件。LSM-100可以扫描文字，也能扫描图像，被扫描到的东西会出现在屏幕上面，并可以保存成各种格式的文件。





西乔
设计师，项目经理。
2006年起携创业团队从事Web技术外包开发及产品咨询顾问。
如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享，
请给西乔发邮件：arthur369@gmail.com